

AD-A270 291 DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

material, and making it available to the public. Responses, including the time for reviewing instructions, searching existing data sources, gathering and reviewing the collection of information, Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

2. REPORT DATE Sep 1993		3. REPORT TYPE AND DATES COVERED THESIS/DISSERTATION	
4. TITLE AND SUBTITLE Cooperative Control of Multiple Space Manipulators		5. FUNDING NUMBERS	
6. AUTHOR(S) Gary E. Yale			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT Student Attending: Naval postgraduate School		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/CI/CIA- 93-20D	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) DEPARTMENT OF THE AIR FORCE AFIT/CI 2950 P STREET WRIGHT-PATTERSON AFB OH 45433-7765		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release IAW 190-1 Distribution Unlimited MICHAEL M. BRICKER, SMSgt, USAF Chief Administration		12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <div style="text-align: center;"><b>DTIC</b> <b>SELECTED</b> <b>OCT 05 1993</b></div> <div style="text-align: right;"><b>93-23213</b> </div>			
14. SUBJECT TERMS		15. NUMBER OF PAGES	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT		18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT
			20. LIMITATION OF ABSTRACT

# NAVAL POSTGRADUATE SCHOOL Monterey, California



## DISSERTATION

COOPERATIVE CONTROL  
OF  
MULTIPLE SPACE MANIPULATORS

by

Gary E. Yale

September, 1993

Dissertation Supervisor:

Brij N. Agrawal

Approved for public release; distribution is unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (If applicable) 55	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS		
		Program Element No	Project No	Task No
				Work Unit Accession Number
11. TITLE (Include Security Classification) COOPERATIVE CONTROL OF MULTIPLE SPACE MANIPULATORS (UNCLASSIFIED)				
12. PERSONAL AUTHOR(S) Yale, Gary E.				
13a. TYPE OF REPORT PhD Dissertation	13b. TIME COVERED From To	14. DATE OF REPORT (year, month, day) September 1993	15. PAGE COUNT 181	
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
17. COSATI CODES			18. SUBJECT TERMS (continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUBGROUP	Space Robotics, Cooperative Control, Attitude Control	
19. ABSTRACT (continue on reverse if necessary and identify by block number) This research concerns the development of cooperative control of two spacecraft mounted two-link manipulators as they reposition a common payload. Lagrangian formulation is used to determine the system equations of motion. Lyapunov stability theory is used to develop the cooperative control by using a reference trajectory and reference actuator torques. Polynomial curves represent potential reference trajectories. Numerical methods select specific reference trajectories to minimize the disturbance torque transmitted to the spacecraft during the payload repositioning maneuver. The reference actuator torques are selected to minimize weighted norms of the torques. Analytical and experimental models of planar motion are used to study the performance of different cooperative controllers. The fifth order polynomial reference trajectory leads to superior performance in terms of spacecraft attitude accuracy, actuator torque magnitude, payload repositioning accuracy, and maneuver time. The higher order polynomial reference trajectory results in only minor improvement in performance. The experimental results verify the concept of cooperative control.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS REPORT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Agrawal, Brij N.			22b. TELEPHONE (Include Area code) (408) 656-3338	22c. OFFICE SYMBOL AA/Ag

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted  
All other editions are obsoleteSECURITY CLASSIFICATION OF THIS PAGE  
Unclassified

Approved for public release; distribution is unlimited.

Cooperative Control of Multiple Space Manipulators

by  
Gary E. Yale  
Captain, United States Air Force  
B.S., United States Air Force Academy, 1981  
M.S., Massachusetts Institute of Technology, 1985

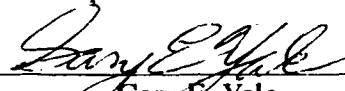
Submitted in partial fulfillment of the  
requirements for the degree of

DOCTOR OF PHILOSOPHY IN  
AERONAUTICAL AND ASTRONAUTICAL ENGINEERING

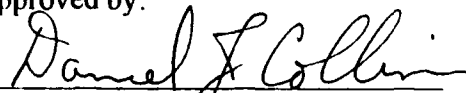
from the


NAVAL POSTGRADUATE SCHOOL  
September, 1993

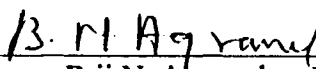
Author: \_\_\_\_\_

  
Gary E. Yale

Approved by: \_\_\_\_\_

  
Daniel J. Collins, Chairman  
Department of Aeronautics & Astronautics

  
I. Michael Ross  
Professor of Aeronautics & Astronautics


  
Brij N. Agrawal  
Professor of Aeronautics & Astronautics  
Dissertation Supervisor

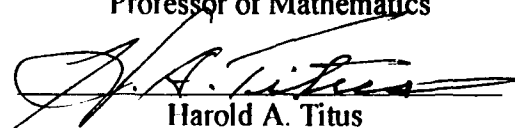
Approved by: \_\_\_\_\_

  
Daniel J. Collins, Chairman, Department of Aeronautics and Astronautics

Approved by: \_\_\_\_\_

Richard S. Elster, Dean of Instruction

  
Donald A. Danielson  
Professor of Mathematics

  
Harold A. Titus  
Professor of Electrical Engineering

## ABSTRACT

This research concerns the development of cooperative control of two spacecraft mounted two-link manipulators as they reposition a common payload. Lagrangian formulation is used to determine the system equations of motion. Lyapunov stability theory is used to develop the cooperative control by using a reference trajectory and reference actuator torques. Polynomial curves represent potential reference trajectories. Numerical methods select specific reference trajectories to minimize the disturbance torque transmitted to the spacecraft during the payload repositioning maneuver. The reference actuator torques are selected to minimize weighted norms of the torques. Analytical and experimental models of planar motion are used to study the performance of different cooperative controllers. The fifth order polynomial reference trajectory leads to superior performance in terms of spacecraft attitude accuracy, actuator torque magnitude, payload repositioning accuracy, and maneuver time. The higher order polynomial reference trajectory results in only minor improvement in performance. The experimental results verify the concept of cooperative control.

Accession For

WILLIAMS, CHARL	<input checked="" type="checkbox"/>
WILLIAMS, CHAR	<input type="checkbox"/>
WILLIAMS, CHAR	<input type="checkbox"/>

A-1

## ACKNOWLEDGMENTS

This work would not have been possible without the contributions from many people. Mr. Rafford Bailey designed the manipulators and toiled numerous hours in AutoCad documenting the design in drawings. Mr. John Moulton converted the drawings into expertly crafted components. Professor John L. Junkins, the George J. Eppright Chair Professor at Texas A&M University and visiting Co-Chair of the Space Systems Academic Group at the Naval Postgraduate School suggested the general scope of this work and ensured successful development of a fixed-base version. Dr. Hyochoong Bang provided significant assistance with the theoretical work. In addition to his guidance regarding the analytical and experimental work, Professor Agrawal was a constant source of encouragement. Finally, the author's family supplied a tremendous amount of support for which he is deeply grateful.

# TABLE OF CONTENTS

I. INTRODUCTION .....	1
A. BACKGROUND AND LITERATURE SURVEY .....	1
B. DISSERTATION OVERVIEW AND OBJECTIVES .....	5
II. ANALYTICAL MODEL .....	7
A. COORDINATE SYSTEMS .....	7
B. EQUATIONS OF MOTION .....	9
1. Inertia Matrix, $M$ .....	11
2. Centripetal and Coriolis Matrix, $G$ .....	17
3. Generalized Forces, $Q$ .....	19
4. Constraints Matrix, $A$ .....	21
C. SIMPLIFIED EQUATIONS OF MOTION .....	23
D. REFERENCE TORQUES .....	24
E. LYAPUNOV CONTROLLER .....	26
F. REFERENCE TRAJECTORIES .....	29
1. Calculating Redundant Coordinates .....	29
2. Selecting Reference Trajectories .....	35
III. VALIDATION AND SIMULATION RESULTS .....	45
A. VALIDATION .....	45
1. Conservation of Kinetic Energy .....	45
2. Conservation of Angular Momentum .....	52
3. Wheel Torque and Constraints .....	58
B. SIMULATIONS .....	60

1. Lyapunov Point Controller .....	60
2. Lyapunov Tracking Controller .....	65
3. Modified Lyapunov Tracking Controller .....	72
4. Comparison of Controllers.....	75
IV. EXPERIMENTAL WORK .....	77
A. SETUP .....	77
1. Centerbody .....	79
2. Manipulators .....	80
3. Payload .....	83
4. Controller .....	83
5. System Integration .....	84
B. RESULTS .....	86
V. SUMMARY AND CONCLUSIONS .....	93
A. SUMMARY .....	93
B. ORIGINAL CONTRIBUTIONS .....	94
C. RECOMMENDATIONS FOR FURTHER STUDY .....	95
APPENDIX A: EXPERIMENTAL CONTROL BLOCK DIAGRAMS .....	97
APPENDIX B: MATLAB CODE .....	109
A. AngMo .....	110
B. AngMo2 .....	112
C. Draw3 .....	114
D. Eqn2 .....	116
E. fminu2 .....	121
F. MainMin .....	126
G. MainOpt .....	129
H. Main2 .....	131



I. Matx .....	142
J. MatxFix .....	146
K. Ref2 .....	150
L. RefMin2 .....	158
REFERENCES .....	165
INITIAL DISTRIBUTION LIST .....	167

## LIST OF FIGURES

Figure 1:	Dual Two-Link Manipulator Configuration .....	8
Figure 2:	Deriving Left Manipulator Joint Angles .....	30
Figure 3:	Deriving Right Manipulator Joint Angles .....	32
Figure 4:	Test Case 1 Angles .....	47
Figure 5:	Test Case 1 Angular Rates .....	47
Figure 6:	Test Case 1 Time Lapse Stick Figure .....	48
Figure 7:	Test Case 1 Kinetic Energy .....	48
Figure 8:	Test Case 1 Angular Momentum .....	49
Figure 9:	Test Case 2 Angles .....	50
Figure 10:	Test Case 2 Angular Rates .....	50
Figure 11:	Test Case 2 Time Lapse Stick Figure .....	51
Figure 12:	Test Case 2 Kinetic Energy .....	51
Figure 13:	Test Case 2 Angular Momentum .....	52
Figure 14:	Test Case 3 Angles .....	53
Figure 15:	Test Case 3 Angular Rates .....	54
Figure 16:	Test Case 3 Time Lapse Stick Figure .....	54
Figure 17:	Test Case 3 Kinetic Energy .....	55
Figure 18:	Test Case 3 Angular Momentum .....	55
Figure 19:	Test Case 4 Angles .....	56
Figure 20:	Test Case 4 Angular Rates .....	56
Figure 21:	Test Case 4 Time Lapse Stick Figure .....	57

Figure 22:	Test Case 4 Kinetic Energy .....	57
Figure 23:	Test Case 4 Angular Momentum .....	58
Figure 24:	Sample of Wheel Torque and Change in Angular Momentum vs. Time .....	59
Figure 25:	Sample of Constraints vs. Time .....	60
Figure 26:	Desired Repositioning Maneuver .....	61
Figure 27:	Lyapunov Point Controller Angles .....	62
Figure 28:	Lyapunov Point Controller Displacement Errors .....	62
Figure 29:	Lyapunov Point Controller Angular Rates .....	63
Figure 30:	Lyapunov Point Controller Command Torques .....	63
Figure 31:	Lyapunov Point Controller Time Lapse Stick Figure .....	64
Figure 32:	Lyapunov Point Controller Initial and Final Stick Figures .....	64
Figure 33:	5th Order Reference Trajectories .....	66
Figure 34:	5th Order Angles .....	66
Figure 35:	5th Order Displacement Errors .....	67
Figure 36:	5th Order Angular Rates .....	67
Figure 37:	5th Order Command Torques .....	68
Figure 38:	5th Order Time Lapse Stick Figure .....	68
Figure 39:	8th Order Reference Trajectories .....	70
Figure 40:	8th Order Angles .....	70
Figure 41:	8th Order Angular Rates .....	71
Figure 42:	8th Order Command Torques .....	71
Figure 43:	8th Order Time Lapse Stick Figure .....	72
Figure 44:	Modified Lyapunov Tracking Controller Angles .....	73
Figure 45:	Modified Lyapunov Tracking Controller Displacement Errors .....	73

Figure 46:	Modified Lyapunov Tracking Controller Angular Rates .....	74
Figure 47:	Modified Lyapunov Tracking Controller Command Torques .....	74
Figure 48:	Modified Lyapunov Tracking Controller Time Lapse Stick Figure .....	75
Figure 49:	Spacecraft Robotics Simulator .....	78
Figure 50:	System Top View .....	79
Figure 51:	Left Manipulator Top and Side Views .....	81
Figure 52:	Right Manipulator Top and Side Views .....	82
Figure 53:	Desired Experimental Repositioning Maneuver .....	86
Figure 54:	$\theta_P$ Commanded, Actual, and Error Angles vs Time .....	87
Figure 55:	$\theta_{L1}$ Commanded, Actual, and Error Angles vs Time .....	88
Figure 56:	$\theta_{L2}$ Commanded, Actual, and Error Angles vs Time .....	89
Figure 57:	$\theta_{R1}$ Commanded, Actual, and Error Angles vs Time .....	90
Figure 58:	$\theta_{R2}$ Commanded, Actual, and Error Angles vs Time .....	91
Figure 59:	Super Blocks Hierarchy .....	97
Figure 60:	Overall Control Block Diagram .....	99
Figure 61:	Parameters Block Diagram .....	100
Figure 62:	Reference Trajectory Block Diagram .....	101
Figure 63:	Encoders Block Diagram .....	102
Figure 64:	Left Angles Block Diagram .....	103
Figure 65:	Part 1 Block Diagram .....	104
Figure 66:	Part 2 Block Diagram .....	105
Figure 67:	Part 3 Block Diagram .....	106
Figure 68:	Right Manipulator Controller Block Diagram .....	107
Figure 69:	Left Manipulator Controller Block Diagram .....	108

Figure 70: MATLAB Modules Hierarchy ..... 109

## LIST OF TABLES

TABLE 1.	GENERIC MODEL SYSTEM PROPERTIES .....	46
TABLE 2.	COMPARISON OF HYPOTHETICAL MODEL SIMULATIONS .....	76
TABLE 3.	MOMENTUM WHEEL MOTOR CHARACTERISTICS .....	80
TABLE 4.	MANIPULATOR ACTUATOR CHARACTERISTICS .....	82
TABLE 5.	POWER SUPPLIES CHARACTERISTICS .....	83
TABLE 6.	EXPERIMENTAL ERROR ANGLES .....	92

# I. INTRODUCTION

## A. BACKGROUND AND LITERATURE SURVEY

Robots are presently an integral part of industrial processes. They perform tasks with high precision, speed and reliability. These same features make robots attractive with regards to space applications.

Space based robotics platforms experience conditions unlike those of their terrestrial counterparts. With respect to the dynamics of the systems, the most notable difference is the absence of a fixed base on which to locate the manipulators. The consequence of the difference is that the motion of the space based manipulator transmits forces and moments to its mounting base resulting in translation and rotation of the base itself. This of course impacts the location of the manipulator's end effector. The problem is further complicated in that the disturbances are not simply a function of the present manipulator joint angles but are also a function of the joint angle histories preceding the current configuration. (Ref. 1)

A number of approaches have been used for dealing with this coupling of joint angle histories and spacecraft main body attitude. Wang (Ref. 2) eliminates the problem by carefully defining what he expects of his dual-arm maneuverable space robot. He prepositions the manipulators such that they are configured to grasp the payload once the vehicle moves within range. After the manipulators are in position, their joints are locked while the spacecraft maneuvers to a location and attitude near the payload. Next, the vehicle approaches the payload in a straight line until the end effectors can grasp the payload. While the manipulator joints remain locked, the vehicle repositions the entire rigid body system to the desired payload destination. At this point, the payload is released and the

vehicle backs away along a straight line. The repositioning of the payload is accomplished by means of the vehicle attitude control rather than altering the joint angles in the manipulators. The manipulators themselves are not moved except when the attitude disturbance they impart to the vehicle is of no importance. Maintaining vehicle attitude during manipulator motion is not a requirement.

Longman, Lindberg and Zedd (Ref. 3) calculate the disturbance torques caused by manipulator motion. This information is used to calculate reaction wheel commands which will compensate for the disturbance torques. In this way, spacecraft attitude control is maintained while the manipulator is repositioned.

If the vehicle does not contain reaction wheels, the primary source of attitude control is probably reaction control thrusters. Because fuel is consumable and hence mission limiting, firing thrusters to hold spacecraft attitude should be avoided whenever possible. Vafa and Dubowsky (Ref. 4) and Longman (Ref. 5) use similar approaches to eliminate the need for reaction thruster firings. Both techniques involve constructing a manipulator trajectory which involves revolving the manipulator in a small coning motion at intermediate stages of the payload repositioning maneuver. This motion imparts a slow rotation of the spacecraft about the coning axis. Careful use of the coning locations permits repositioning of the payload between any arbitrary locations (within manipulator reach) and attitudes while also changing the spacecraft to any desired attitude without the need for thruster firings. This technique does not, however, maintain a particular spacecraft attitude during the maneuver.

Nakamura and Mukherjee (Ref. 6) use a technique called the bi-directional approach. This method represents a six degree of freedom (DOF) manipulator mounted on a space vehicle as a nine variable system (six joint angles and three spacecraft attitude angles) with six inputs (the manipulator joint torques). They attack the problem from both ends by integrating forward from the initial conditions and backwards from the desired final



conditions. A Lyapunov function guarantees that the two solutions will converge at some intermediate time during the maneuver. As in Ref. 4 and Ref. 5, the payload is repositioned to the desired location and attitude and the attitude of the spacecraft main body is changed to its desired orientation. However, the main body attitude during the maneuver is not controlled. Furthermore, the joint angle trajectories calculated by this method contain rapid, large oscillations near the maneuver start and stop times and noncontinuous derivatives at the convergence time.

Vafa (Ref. 7) succeeds in using a single space-based manipulator to control spacecraft attitude during a repositioning maneuver. He does this by employing a nine DOF manipulator. This manipulator has enough redundancy in its kinematics to control the end effector location and attitude and the spacecraft attitude. Six DOF are allocated to repositioning the payload and the remaining three DOF are used to control the spacecraft attitude.

Like Ref. 7, the primary objective of Chung, Desa and deSilva (Ref. 8) is to address the disturbances transmitted to the spacecraft by the manipulator motion. They also use a single manipulator with redundant kinematics. Because they use inverse kinematics to find the joint torques, the manipulator redundancy prevents the existence of a unique solution. A solution is selected from among the infinity of possible solutions by means of minimizing a cost function of the magnitudes of the forces and torques transmitted to the base.

Torres and Dubowsky (Ref. 9) also focus on the spacecraft attitude disturbances caused by manipulator motion. They recognize that for any given point in joint space, there is a direction of motion which produces minimum spacecraft attitude disturbance and a perpendicular direction of motion which produces maximum spacecraft attitude disturbance. A tool called the enhanced disturbance map (EDM) depicts these directions graphically. The EDM permits users to plan manipulator trajectories that lie on or near

zero disturbance paths. For non redundant manipulators, this may involve repositioning the spacecraft itself prior to the manipulator repositioning maneuver. If the manipulator has redundant kinematics, one can find a zero disturbance path to connect the manipulator trajectory endpoints without having to reposition the spacecraft initially. This technique considers only the location of the manipulator endpoint and not its attitude.

Configurations with multiple manipulators have also been explored. The closed chain nature of these configurations prevent the use of some of the techniques already discussed. In addition, controlling multiple manipulators raises the issue of cooperation between the manipulators.

Nguyen, Pooran and Premack (Ref. 10) develop a PD controller for a fixed base, two DOF, closed chain manipulator system. The system is linearized by means of Taylor series expansion about a point designated as the robot's "home" point. Pole placement is then used to select controller gains.

Hu and Goldenberg (Ref. 11) derive an adaptive control scheme for multiple nonredundant manipulators mounted to a fixed base. In this reference, coordinated control involves controlling the motion of the grasped object, the contact forces between the object and its environment, and the internal forces within the object caused from being held by more than one manipulator.

For a space based system, contact forces between the payload and its environment are less likely to be important. Walker, Kim and Dionise (Ref. 12) present just such an adaptive controller.

Coordinated control of multiple manipulators assumes a different meaning according to Yoshida, Kurazume and Umetani (Ref. 13). Although they propose a system with two manipulators, only one actually grasps the payload. The other is used to provide counterbalancing torques to the spacecraft main body. The role of the second manipulator is sim-

ilar to a reaction wheel in that its primary function is to control spacecraft attitude rather than reposition a payload.

Ahmad and Zribi (Ref. 14) apply a Lyapunov controller to a fixed base, multiple manipulator system. As in Ref. 12, they are concerned with controlling the payload position and its internal forces. To do so, the method requires sensors to measure the forces and moments created by each manipulator. They also present an adaptive version to control this system.

While still addressing payload position and internal forces, Schneider and Cannon (Ref. 15) use a technique called object impedance control to achieve coordinated control among the manipulators. This method views the payload as being anchored to a desired location by a spring/damper system.

## **B. DISSERTATION OVERVIEW AND OBJECTIVES**

This research is concerned with the cooperative control of a space based manipulator system with multiple manipulators handling a common payload. The scope is limited to planar motion in which the spacecraft is allowed to rotate but not to translate. These restrictions permit experimental verification in the Spacecraft Dynamics and Control Laboratory at the Naval Postgraduate School. The objectives of this research are to 1) develop a stable control law which facilitates cooperation among the manipulators as they reposition the payload, 2) minimize joint actuator effort, 3) reduce the disturbance torque transmitted to the spacecraft main body by the manipulator motion, and 4) validate the analytical development with experimental results.

Chapter II develops the analytical model in detail. Coordinate systems are defined and the equations of motion are derived. A technique for finding control torques which minimizes a weighted norm is presented. A globally stable control law is developed using

Lyapunov methods. The idea of using a reference trajectory to describe the motion is applied as are methods for choosing the reference trajectory.

Chapter III verifies the analytical model with several test cases. The model is evaluated for compliance with the principles of conservation of kinetic energy and angular momentum. After establishing the validity of the model, results from simulations are presented. The stability of the controller is illustrated as is the dramatic improvement in performance when a reference trajectory is included. Results from a simplified control law which is more practical to implement are included and compared to the complete control law version.

Discussion of the experimental work is contained in Chapter IV. This chapter includes a description of the experimental setup. As might be expected, actual hardware demonstrated that there are differences between the ideal world of the analytical model and the real world of hardware implementation.

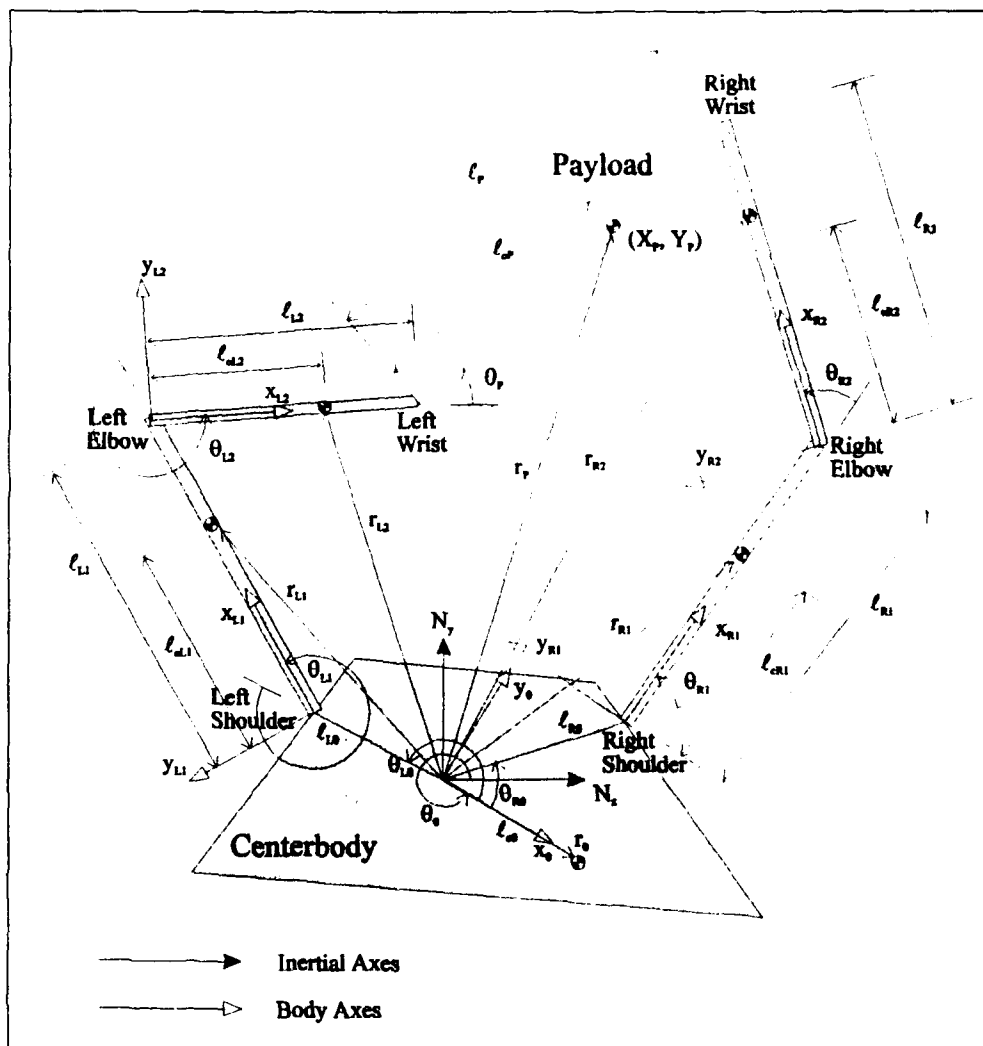
The summary and concluding remarks are presented in Chapter V. This chapter also contains suggestions for future work in this field.

## II. ANALYTICAL MODEL

The analytical model represents a spacecraft with two manipulators attached. The manipulators have already firmly grasped an object hereafter referred to as the payload. The manipulators are about to reposition the payload with respect to the spacecraft. The ensuing dynamics between the spacecraft, manipulators, and payload are the topic of this research. What occurs before the manipulators grasp the payload and after they release it is beyond the scope of this investigation. The scope is narrowed further by confining the motion to be two dimensional and allowing the spacecraft to rotate but not translate. These assumptions are consistent with hardware restrictions during experimental verification.

### A. COORDINATE SYSTEMS

Figure 1 shows a schematic of the overall system. This diagram illustrates the relationship between the coordinate frames used to develop the equations of motion. All angles are measured positive counterclockwise. The centerbody, manipulator links, and payload are assumed to be rigid bodies. Therefore, member lengths ( $\ell_{L1}$ ,  $\ell_{L2}$ ,  $\ell_{R1}$ ,  $\ell_{R2}$ , and  $\ell_P$ ), distances to member centers of mass ( $\ell_{c0}$ ,  $\ell_{cL1}$ ,  $\ell_{cL2}$ ,  $\ell_{cR1}$ ,  $\ell_{cR2}$ , and  $\ell_{cP}$ ), and the location of the left and right shoulders ( $\ell_{L0}$ ,  $\theta_{L0}$ ,  $\ell_{R0}$ , and  $\theta_{R0}$ ) remain constant. An inertial axis system is located on the centerbody at the point of rotation. A body fixed coordinate frame uses the same origin as the inertial frame but rotates with the spacecraft centerbody. The x-axis of this frame points to the centerbody center of mass. The centerbody attitude,  $\theta_0$ , is the angle between the inertial x-axis and the spacecraft centerbody x-axis. Each manipulator link has its own set of body axes. A coordinate frame attached to the left



**Figure 1: Dual Two-Link Manipulator Configuration**

shoulder aligns its x-axis along the longitudinal axis of manipulator Link L1. The attitude of this link,  $\theta_{L1}$ , is zero when the link lies on a ray extending from the inertial origin through the left shoulder. The attitude of Link L2 is defined by a coordinate frame attached to the left elbow. The attitude of this link,  $\theta_{L2}$ , is zero when the link is parallel with the proceeding link, Link L1. Similar coordinate frames and definitions apply to the right manipulator. The payload attitude,  $\theta_p$ , is referenced to the inertial frame. The Cartesian coordinates of the payload center of mass are also with respect to the inertial frame. A set of generalized coordinates which describe the system include the centerbody attitude, joint angles for all of the manipulator links, and payload attitude and position.

$$\underline{q} = [\theta_0 \theta_{L1} \theta_{L2} \theta_{R1} \theta_{R2} \theta_p X_p Y_p]^T \quad (1)$$

Six joint actuators apply torques at the shoulder, elbow, and wrist of each manipulator. A reaction wheel applies a torque to the centerbody. The actuators can be grouped into a control vector

$$\underline{u} = [u_{wh} u_{LS} u_{LE} u_{LW} u_{RS} u_{RE} u_{RW}]^T \quad (2)$$

where the first element is the reaction wheel torque. The remaining elements are joint actuator torques. The first letter of their torque subscripts indicates left or right arm. The second subscript indicates shoulder (S), elbow (E) or wrist (W).

## B. EQUATIONS OF MOTION

The equations of motion for this system are developed using Lagrange's equations for a dynamic system with holonomic constraints.

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}} \right) - \frac{\partial L}{\partial q} = \underline{Q} + \underline{A}^T \underline{\lambda} \quad (3)$$

subject to constraints

$$A\dot{q} + A_0 = 0 \quad (4)$$

where  $L = T - V$

$T$  is kinetic energy

$V$  is potential energy

$q$  is the generalized coordinates vector

$\dot{q}$  is the generalized velocities vector

$Q$  is the vector of applied nonconservative forces

$A^T \lambda$  are the constraint forces

Beginning with Lagrange's equation, the equations of motion can be rearranged into an alternate form. The inertia matrix,  $M$ , is a function of the generalized coordinates. Since the potential energy is a function of only the generalized coordinates, the partial of the Lagrangian with respect to the generalized velocities does not contain any potential energy terms.

$$\frac{\partial L}{\partial \dot{q}} = M\dot{q} \quad (5)$$

Differentiating Eq. (5) with respect to time leads to

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}} \right) = M\ddot{q} + \dot{q}^T \frac{dM}{dt} \dot{q} = M\ddot{q} + \dot{q}^T \frac{\partial M}{\partial q} \dot{q} \quad (6)$$

Replacing the Lagrangian with expressions for kinetic energy and potential energy results in

$$\frac{\partial L}{\partial q} = \frac{1}{2} \left( \dot{q}^T \frac{\partial M}{\partial q} \dot{q} \right) - \frac{\partial V}{\partial q} \quad (7)$$

Eq. (6) and Eq. (7) can be substituted back into Eq. (3) to produce

$$M\ddot{q} + \frac{1}{2} \left( \dot{q}^T \frac{\partial M}{\partial q} \dot{q} \right) + \frac{\partial V}{\partial q} = Q + A^T \lambda \quad (8)$$



The second term on the left-hand side of Eq. (8) contains the centripetal and Coriolis torques. Replacing this with the G matrix leads to

$$M\ddot{q} + G(q, \dot{q}) + \frac{\partial V}{\partial q} = Q + A^T \lambda \quad (9)$$

After substituting the matrix form of the generalized forces into the equations of motion, one has

$$M\ddot{q} + G + \frac{\partial V}{\partial q} = Bu + A^T \lambda \quad (10)$$

The following sections develop expressions for the inertia matrix, Coriolis and centripetal accelerations, generalized forces, and constraints imposed by the closed chain geometry of the system.

### 1. Inertia Matrix, M

The inertia matrix is found by calculating the system kinetic energy and expressing it in the form

$$T = \frac{1}{2} \dot{q}^T [M(q)] \dot{q} \quad (11)$$

The inertia matrix is the term bracketed by the generalized velocity vectors. The total system kinetic energy is the sum of the kinetic energy of all the pieces.

$$T = T_0 + T_{L1} + T_{L2} + T_{R1} + T_{R2} + T_P \quad (12)$$

Kinetic energy of individual components can be found from

$$T_i = \frac{1}{2} I_i \omega_i^2 + \frac{1}{2} m_i (\dot{r} \cdot \dot{r}) \quad (13)$$

$I_i$  is the member moment of inertia about its center of mass

$\omega_i$  is the angular velocity

$m_i$  is the mass

$\dot{r}$  is the inertial velocity of the center of mass

The centerbody angular rate and center of mass position vector are given by

$$\omega_0 = \dot{\theta}_0 \quad (14)$$

$$r_0 = \ell_{c0} \hat{s}_0 \quad (15)$$

Differentiating Eq. (15) results in the velocity of the centerbody center of mass

$$\dot{r}_0 = \ell_{c0} \omega_0 \hat{s}_0 \quad (16)$$

Substituting Eq. (14) and Eq. (16) back into the expression for kinetic energy (Eq. (13)) and collecting on the angular rate term leads to

$$T_0 = \frac{1}{2} (I_0 + m_0 \ell_{c0}^2) \dot{\theta}_0^2 \quad (17)$$

Similar developments are used for each of the remaining pieces in the system. For the left manipulator link between the shoulder and elbow (Link L1), the angular velocity is a combination of centerbody rotation and rotation of the link with respect to the centerbody.

$$\omega_{L1} = \dot{\theta}_0 + \dot{\theta}_{L1} \quad (18)$$

The position vector to the center of mass is

$$r_{L1} = \ell_{L0} \cos \theta_{L0} \hat{s}_0 + \ell_{L0} \sin \theta_{L0} \hat{y}_0 + \ell_{cL1} \hat{s}_{L1} \quad (19)$$

The first two terms in the position vector represent the location of the left shoulder. Differentiating the position vector gives the expression for the velocity vector. Because none of the coordinate axes used in the position vector expression are inertial, their rotation must be included as well.

$$\dot{r}_{L1} = \ell_{L0} \omega_0 \cos \theta_{L0} \hat{y}_0 - \ell_{L0} \omega_0 \sin \theta_{L0} \hat{s}_0 + \ell_{cL1} \omega_{L1} \hat{s}_{L1} \quad (20)$$

After Eqs. (18) and (20) are substituted into Eq. (13) and terms are grouped by angular rates, the kinetic energy is

$$T_{L1} = \dot{\theta}_0^2 (0.5 (I_{L1} + m_{L1} \ell_{cL1}^2 + m_{L1} \ell_{L0}^2)) \quad (21)$$

$$\begin{aligned}
& + m_{L1} \ell_{L0} \ell_{cL1} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1}) ) ) \\
& + 0.5 \dot{\theta}_{L1}^2 (I_{L1} + m_{L1} \ell_{cL1}^2) \\
& + \dot{\theta}_0 \dot{\theta}_{L1} (I_{L1} + m_{L1} \ell_{cL1}^2 + m_{L1} \ell_{L0} \ell_{cL1} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1}) ) )
\end{aligned}$$

The angular rate for the left forearm includes the centerbody angular rate as well as the angular rates of the body axes on Links L1 and L2.

$$\omega_{L2} = \dot{\theta}_0 + \dot{\theta}_{L1} + \dot{\theta}_{L2} \quad (22)$$

This link's center of mass position vector is

$$r_{L2} = \ell_{L0} \cos \theta_{L0} \hat{x}_0 + \ell_{L0} \sin \theta_{L0} \hat{y}_0 + \ell_{L1} \hat{x}_{L1} + \ell_{cL2} \hat{x}_{L2} \quad (23)$$

Differentiating the position vector gives the velocity vector.

$$\dot{r}_{L2} = \ell_{L0} \omega_0 \cos \theta_{L0} \hat{y}_0 - \ell_{L0} \omega_0 \sin \theta_{L0} \hat{x}_0 + \ell_{L1} \omega_{L1} \hat{y}_{L1} + \ell_{cL2} \omega_{L2} \hat{y}_{L2} \quad (24)$$

The kinetic energy expression for Link L2 is found after substituting Eqs. (22) and (24) into Eq. (13) and collecting terms with common angular rates.

$$\begin{aligned}
T_{L2} = & \dot{\theta}_0^2 (0.5 (I_{L2} + m_{L2} \ell_{cL2}^2 + m_{L2} \ell_{L1}^2 + m_{L2} \ell_{L0}^2) \\
& + m_{L2} \ell_{L0} \ell_{L1} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1}) ) + m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2} \\
& + m_{L2} \ell_{L0} \ell_{cL2} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1} + \theta_{L2}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1} + \theta_{L2}) ) ) \\
& + \dot{\theta}_{L1}^2 (0.5 (I_{L2} + m_{L2} \ell_{L1}^2 + m_{L2} \ell_{cL2}^2) + m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2}) \\
& + 0.5 \dot{\theta}_{L2}^2 (I_{L2} + m_{L2} \ell_{cL2}^2) \\
& + \dot{\theta}_0 \dot{\theta}_{L1} (I_{L2} + m_{L2} \ell_{L1}^2 + m_{L2} \ell_{cL2}^2 + 2 m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2} \\
& + m_{L2} \ell_{L0} \ell_{L1} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1}) ) \\
& + m_{L2} \ell_{L0} \ell_{cL2} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1} + \theta_{L2}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1} + \theta_{L2}) ) ) \\
& + \dot{\theta}_0 \dot{\theta}_{L2} (I_{L2} + m_{L2} \ell_{cL2}^2 + m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2} \\
& + m_{L2} \ell_{L0} \ell_{cL2} ( \sin \theta_{L0} \sin (\theta_{L0} + \theta_{L1} + \theta_{L2}) + \cos \theta_{L0} \cos (\theta_{L0} + \theta_{L1} + \theta_{L2}) ) )
\end{aligned} \quad (25)$$

$$+ \dot{\theta}_{L1} \dot{\theta}_{L2} (I_{L2} + m_{L2} \ell_{cL2}^2 + m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2})$$

The development for the right manipulator kinetic energy parallels that for the left manipulator. For the upper arm portion (Link R1), the angular rate is

$$\omega_{R1} = \dot{\theta}_0 + \dot{\theta}_{R1} \quad (26)$$

The position vector is constructed by finding the coordinates of the right shoulder and adding the vector from the shoulder to the center of mass.

$$\mathbf{r}_{R1} = \ell_{R0} \cos \theta_{R0} \hat{x}_0 + \ell_{R0} \sin \theta_{R0} \hat{y}_0 + \ell_{cR1} \hat{x}_{R1} \quad (27)$$

The time rate of change of the position vector is

$$\dot{\mathbf{r}}_{R1} = \ell_{R0} \omega_0 \cos \theta_{R0} \hat{y}_0 - \ell_{R0} \omega_0 \sin \theta_{R0} \hat{x}_0 + \ell_{cR1} \omega_{R1} \hat{y}_{R1} \quad (28)$$

After calculating the kinetic energy for Link R1 and grouping terms with common angular rates, the resulting expression is

$$T_{R1} = \dot{\theta}_0^2 (0.5 (I_{R1} + m_{R1} \ell_{cR1}^2 + m_{R1} \ell_{R0}^2) \quad (29)$$

$$+ m_{R1} \ell_{R0} \ell_{cR1} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1})) )$$

$$+ 0.5 \dot{\theta}_{R1}^2 (I_{R1} + m_{R1} \ell_{cR1}^2)$$

$$+ \dot{\theta}_0 \dot{\theta}_{R1} (I_{R1} + m_{R1} \ell_{cR1}^2 + m_{R1} \ell_{R0} \ell_{cR1} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1})))$$

Angular rate of the right forearm is

$$\omega_{R2} = \dot{\theta}_0 + \dot{\theta}_{R1} + \dot{\theta}_{R2} \quad (30)$$

Its position vector is

$$\mathbf{r}_{R2} = \ell_{R0} \cos \theta_{R0} \hat{x}_0 + \ell_{R0} \sin \theta_{R0} \hat{y}_0 + \ell_{R1} \hat{x}_{R1} + \ell_{cR2} \hat{x}_{R2} \quad (31)$$

Differentiating Eq. (31) produces the velocity vector for Link R2 center of mass.

$$\dot{\mathbf{r}}_{R2} = \ell_{R0} \omega_0 \cos \theta_{R0} \hat{y}_0 - \ell_{R0} \omega_0 \sin \theta_{R0} \hat{x}_0 + \ell_{R1} \omega_{R1} \hat{y}_{R1} + \ell_{cR2} \omega_{R2} \hat{y}_{R1} \quad (32)$$

The kinetic energy resulting from substituting Eqs. (30) and (32) into Eq. (13) and collecting common angular rate terms is

$$\begin{aligned}
 T_{R2} = & \dot{\theta}_0^2 (0.5 (I_{R2} + m_{R2} \ell_{cR2}^2 + m_{R2} \ell_{R1}^2 + m_{R2} \ell_{R0}^2) \\
 & + m_{R2} \ell_{R0} \ell_{R1} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1})) + m_{R2} \ell_{R1} \ell_{cR2} \cos \theta_{R2} \\
 & + m_{R2} \ell_{R0} \ell_{cR2} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1} + \theta_{R2}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1} + \theta_{R2})) ) \\
 & + \dot{\theta}_{R1}^2 (0.5 (I_{R2} + m_{R2} \ell_{R1}^2 + m_{R2} \ell_{cR2}^2) + m_{R2} \ell_{R1} \ell_{cR2} \cos \theta_{R2}) \\
 & + 0.5 \dot{\theta}_{R2}^2 (I_{R2} + m_{R2} \ell_{cR2}^2) \\
 & + \dot{\theta}_0 \dot{\theta}_{R1} (I_{R2} + m_{R2} \ell_{R1}^2 + m_{R2} \ell_{cR2}^2 + 2 m_{R2} \ell_{R1} \ell_{cR2} \cos \theta_{R2} \\
 & + m_{R2} \ell_{R0} \ell_{R1} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1})) \\
 & + m_{R2} \ell_{R0} \ell_{cR2} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1} + \theta_{R2}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1} + \theta_{R2})) ) \\
 & + \dot{\theta}_0 \dot{\theta}_{R2} (I_{R2} + m_{R2} \ell_{cR2}^2 + m_{R2} \ell_{R1} \ell_{cR2} \cos \theta_{R2} \\
 & + m_{R2} \ell_{R0} \ell_{cR2} (\sin \theta_{R0} \sin (\theta_{R0} + \theta_{R1} + \theta_{R2}) + \cos \theta_{R0} \cos (\theta_{R0} + \theta_{R1} + \theta_{R2})) ) \\
 & + \dot{\theta}_{R1} \dot{\theta}_{R2} (I_{R2} + m_{R2} \ell_{cR2}^2 + m_{R2} \ell_{R1} \ell_{cR2} \cos \theta_{R2})
 \end{aligned} \tag{33}$$

Expressions for the payload are considerably simpler because inertial coordinates are available. The angular rate is

$$\omega_p = \dot{\theta}_p \tag{34}$$

It is not necessary to describe the payload center of mass by passing through either shoulder as was the case with the manipulator links. The position vector is

$$r_p = X_p \hat{N}_x + Y_p \hat{N}_y \tag{35}$$

The velocity vector is also simpler because of the inertial frame.

$$\dot{r}_p = \dot{X}_p \hat{N}_x + \dot{Y}_p \hat{N}_y \tag{36}$$

The payload kinetic energy is derived from substituting Eqs. (14) and (16) into Eq. (13).

$$T_p = \frac{1}{2} (I_p \dot{\theta}_p^2 + m_p (\dot{X}_p^2 + \dot{Y}_p^2)) \quad (37)$$

After substituting the expressions for kinetic energy from Eqs. (17), (21), (25), (29), (33) and (37) into Eq. (12) and expressing the result in the matrix form of Eq. (3), the inertia matrix,  $M$ , is given by Eq. (38). Because the generalized coordinates for the payload are referenced to an inertial coordinate frame, the inertia matrix is decoupled between the payload and the rest of the system. Coupling does exist between the spacecraft center-body and each of the manipulators.

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & 0 & 0 & 0 \\ M_{21} & M_{22} & M_{23} & 0 & 0 & 0 & 0 & 0 \\ M_{31} & M_{32} & M_{33} & 0 & 0 & 0 & 0 & 0 \\ M_{41} & 0 & 0 & M_{44} & M_{45} & 0 & 0 & 0 \\ M_{51} & 0 & 0 & M_{54} & M_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_p & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & m_p & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_p \end{bmatrix} \quad (38)$$

Expressions for the individual elements in the inertia matrix are given by

$$M_{55} = I_{R2} + m_{R2} \ell_{R2}^2 \quad (39)$$

$$M_{45} = M_{54} = M_{55} + m_{R2} \ell_{R1} \ell_{R2} \cos \theta_{R2} \quad (40)$$

$$M_{15} = M_{51} = M_{45} + m_{R2} \ell_{R0} \ell_{R2} \cos (\theta_{R1} + \theta_{R2}) \quad (41)$$

$$M_{44} = M_{45} + I_{R1} + m_{R2} \ell_{R1} \ell_{R2} \cos \theta_{R2} + m_{R1} \ell_{R1}^2 + m_{R2} \ell_{R1}^2 \quad (42)$$

$$M_{14} = M_{41} = M_{44} + \ell_{R0} (m_{R1} \ell_{cR1} + m_{R2} \ell_{cR1}) \cos \theta_{R1} \quad (43)$$

$$+ m_{R2} \ell_{R0} \ell_{cR2} \cos (\theta_{R1} + \theta_{R2})$$

$$M_{33} = I_{L2} + m_{L2} \ell_{L2}^2 \quad (44)$$

$$M_{23} = M_{32} = M_{33} + m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2} \quad (45)$$

$$M_{13} = M_{31} = M_{23} + m_{L2} \ell_{L0} \ell_{cL2} \cos (\theta_{L1} + \theta_{L2}) \quad (46)$$

$$M_{22} = M_{23} + I_{L1} + m_{L2} \ell_{L1} \ell_{cL2} \cos \theta_{L2} + m_{L1} \ell_{cL1}^2 + m_{L2} \ell_{L1}^2 \quad (47)$$

$$M_{12} = M_{21} = M_{22} + \ell_{L0} (m_{L1} \ell_{cL1} + m_{L2} \ell_{cL1}) \cos \theta_{L1} \quad (48)$$

$$+ m_{L2} \ell_{L0} \ell_{cL2} \cos (\theta_{L1} + \theta_{L2})$$

$$M_{11} = I_0 + M_{22} + M_{44} + m_0 \ell_{c0}^2 + (m_{L1} + m_{L2}) \ell_{L0}^2 + (m_{R1} + m_{R2}) \ell_{R0}^2 \quad (49)$$

$$+ 2 \ell_{R0} (m_{R1} \ell_{cR1} + m_{R2} \ell_{cR1}) \cos \theta_{R1} + 2 m_{R2} \ell_{R0} \ell_{cR2} \cos (\theta_{R1} + \theta_{R2})$$

$$+ 2 \ell_{L0} (m_{L1} \ell_{cL1} + m_{L2} \ell_{cL1}) \cos \theta_{L1} + 2 m_{L2} \ell_{L0} \ell_{cL2} \cos (\theta_{L1} + \theta_{L2})$$

## 2. Centripetal and Coriolis Matrix, G

The G matrix contains all of the centripetal and Coriolis terms. It is most easily found using

$$G(\underline{q}, \underline{\dot{q}}) = \begin{bmatrix} \underline{\dot{q}}^T C^{(1)} \underline{\dot{q}} \\ \underline{\dot{q}}^T C^{(2)} \underline{\dot{q}} \\ \dots \\ \underline{\dot{q}}^T C^{(8)} \underline{\dot{q}} \end{bmatrix} \quad (50)$$

where the elements of  $C^{(i)}$  are defined by the Christoffel symbol

$$C_{jk}^{(i)} = \frac{1}{2} \left( \frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right) \quad (51)$$

The form of the G matrix for the system of Figure 1 is given as

$$G = \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (52)$$

$$G_1 = -\ell_{L0} (\dot{\theta}_{L1}^2 + 2\dot{\theta}_0 \dot{\theta}_{L1}) (m_{L1} \ell_{L1} + m_{L2} \ell_{L1}) \sin \theta_{L1} \quad (53)$$

$$-m_{L2} \ell_{L1} \ell_{L2} \dot{\theta}_{L2} (2\dot{\theta}_0 + 2\dot{\theta}_{L1} + \dot{\theta}_{L2}) \sin \theta_{L2}$$

$$-m_{L2} \ell_{L0} \ell_{L2} (2\dot{\theta}_0 (\dot{\theta}_{L1} + \dot{\theta}_{L2}) + (\dot{\theta}_{L1} + \dot{\theta}_{L2})^2) \sin (\theta_{L1} + \theta_{L2})$$

$$-\ell_{R0} (\dot{\theta}_{R1}^2 + 2\dot{\theta}_0 \dot{\theta}_{R1}) (m_{R1} \ell_{R1} + m_{R2} \ell_{R1}) \sin \theta_{R1}$$

$$-m_{R2} \ell_{R1} \ell_{R2} \dot{\theta}_{R2} (2\dot{\theta}_0 + 2\dot{\theta}_{R1} + \dot{\theta}_{R2}) \sin \theta_{R2}$$

$$-m_{R2} \ell_{R0} \ell_{R2} (2\dot{\theta}_0 (\dot{\theta}_{R1} + \dot{\theta}_{R2}) + (\dot{\theta}_{R1} + \dot{\theta}_{R2})^2) \sin (\theta_{R1} + \theta_{R2})$$

$$G_2 = \ell_{L0} \dot{\theta}_0^2 (m_{L1} \ell_{L1} + m_{L2} \ell_{L1}) \sin \theta_{L1} - m_{L2} \ell_{L1} \ell_{L2} \dot{\theta}_{L2} (2\dot{\theta}_0 + 2\dot{\theta}_{L1} + \dot{\theta}_{L2}) \sin \theta_{L2} \quad (54)$$

$$+ m_{L2} \ell_{L0} \ell_{L2} \dot{\theta}_0^2 \sin (\theta_{L1} + \theta_{L2})$$

$$G_3 = m_{L2} \ell_{L1} \ell_{L2} (\dot{\theta}_{L1} + \dot{\theta}_{L2})^2 \sin \theta_{L2} + m_{L2} \ell_{L0} \ell_{L2} \dot{\theta}_0^2 \sin (\theta_{L1} + \theta_{L2}) \quad (55)$$

$$G_4 = \ell_{R0} \dot{\theta}_0^2 (m_{R1} \ell_{R1} + m_{R2} \ell_{R1}) \sin \theta_{R1} - m_{R2} \ell_{R1} \ell_{R2} \dot{\theta}_{R2} (2\dot{\theta}_0 + 2\dot{\theta}_{R1} + \dot{\theta}_{R2}) \sin \theta_{R2} \quad (56)$$

$$+ m_{R2} \ell_{R0} \ell_{R2} \dot{\theta}_0^2 \sin (\theta_{R1} + \theta_{R2})$$

$$G_5 = m_{R2} \ell_{R1} \ell_{R2} (\dot{\theta}_{R1} + \dot{\theta}_{R2})^2 \sin \theta_{R2} + m_{R2} \ell_{R0} \ell_{R2} \dot{\theta}_0^2 \sin (\theta_{R1} + \theta_{R2}) \quad (57)$$



### 3. Generalized Forces, Q

The generalized forces are found using the principle of virtual work. When there is no reaction wheel on the centerbody, the system does not experience any external forces capable of performing work. Six joint actuators apply torques at the shoulder, elbow, and wrist of each manipulator.

$$u_G = [u_{LS} \ u_{LE} \ u_{LW} \ u_{RS} \ u_{RE} \ u_{RW}]^T \quad (58)$$

$u_G$  is simply the joint actuator subset of the complete actuator torque vector,  $u$ . The total virtual work is the sum of the torques applied to the individual bodies times their virtual angular displacements.

$$\delta W = \sum_{i=1}^N \delta W_i = \sum_{i=1}^N (M_i) \delta \theta_i \quad (59)$$

When the left shoulder joint actuator applies a positive torque on Link L1, a negative torque is also applied to the centerbody. The virtual work performed by the left shoulder motor is

$$\delta W_{LS} = u_{LS} (\delta \theta_0 + \delta \theta_{L1} - \delta \theta_0) \quad (60)$$

where the positive angles are those associated with the change in Link L1 attitude and the negative angle is associated with the change in centerbody attitude. The left elbow actuator makes a positive contribution to Link L2 attitude and a negative contribution to Link L1 attitude.

$$\delta W_{LE} = u_{LE} (\delta \theta_0 + \delta \theta_{L1} + \delta \theta_{L2} - \delta \theta_0 - \delta \theta_{L1}) = u_{LE} \delta \theta_{L2} \quad (61)$$

The joint actuator at the left wrist makes a positive change in the payload attitude and a negative change in Link L2.

$$\delta W_{LW} = u_{LW} (\delta \theta_P - \delta \theta_0 - \delta \theta_{L1} - \delta \theta_{L2}) \quad (62)$$

The right shoulder actuator makes a positive contribution to Link R1 attitude and a negative contribution to centerbody attitude.

$$\delta W_{RS} = u_{RS} (\delta\theta_0 + \delta\theta_{R1} - \delta\theta_0) = u_{RS} \delta\theta_{R1} \quad (63)$$

Link R2 has a positive virtual displacement due to a positive torque at the right elbow. The same torque causes a negative virtual displacement of Link R1.

$$\delta W_{RE} = u_{RE} (\delta\theta_0 + \delta\theta_{R1} + \delta\theta_{R2} - \delta\theta_0 - \delta\theta_{R1}) = u_{RE} \delta\theta_{R2} \quad (64)$$

The right wrist actuator has a positive influence on the payload and a negative influence on Link R2.

$$\delta W_{RW} = u_{RW} (\delta\theta_P - \delta\theta_0 - \delta\theta_{R1} - \delta\theta_{R2}) \quad (65)$$

Gathering Eqs. (60)-(65) together produces

$$\begin{aligned} \delta W = & (-u_{LW} - u_{RW}) \delta\theta_0 + (u_{LS} - u_{LW}) \delta\theta_{L1} + (u_{LE} - u_{LW}) \delta\theta_{L2} \\ & + (u_{RS} - u_{RW}) \delta\theta_{R1} + (u_{RE} - u_{RW}) \delta\theta_{R2} + (u_{LW} - u_{RW}) \delta\theta_P \end{aligned} \quad (66)$$

With respect to the system equations of motion, the generalized force corresponding to a particular generalized coordinate is that portion of the virtual work associated with the same generalized coordinate. Now Eq. (66) can be transformed into a matrix form.

$$\underline{Q}_6 = \underline{B}_6 \underline{u}_6 \quad (67)$$

where  $\underline{B}$  is the control influence matrix given by

$$\underline{B}_6 = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (68)$$

The only effect of a positive reaction wheel torque applied to the spacecraft is to alter the centerbody attitude in a positive direction. This manifests itself in the control influence matrix in the form of another column. This new column is all zeros except for a single one corresponding to the location of the reaction wheel torque in the  $\mathbf{u}$  vector. With the reaction wheel torque as the first element in the control vector as in Eq. (2), the complete control influence matrix is

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (69)$$

#### 4. Constraints Matrix, A

Because the eighth order system under consideration has only four degrees of freedom, an additional four equations are needed to describe the constraints. The eight generalized coordinates are not independent. The constraint equations embody the information that the manipulators are both grasping the payload forming a closed chain system. The constraints matrix is derived by writing the system constraints in the Pfaffian form as

$$\mathbf{A}\dot{\mathbf{q}} + \mathbf{A}_0 = 0 \quad (70)$$

These equations come from geometric relationships of expressing the payload center of mass Cartesian coordinates in terms of the other generalized coordinates.

$$\left. \begin{aligned} \ell_{L0} \cos(\theta_0 + \theta_{L0}) + \ell_{L1} \cos(\theta_0 + \theta_{L0} + \theta_{L1}) + \ell_{L2} \cos(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) + \ell_{cP} \cos \theta_P &= x_P \\ \ell_{L0} \sin(\theta_0 + \theta_{L0}) + \ell_{L1} \sin(\theta_0 + \theta_{L0} + \theta_{L1}) + \ell_{L2} \sin(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) + \ell_{cP} \sin \theta_P &= y_P \\ \ell_{R0} \cos(\theta_0 + \theta_{R0}) + \ell_{R1} \cos(\theta_0 + \theta_{R0} + \theta_{R1}) + \ell_{R2} \cos(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) - (\ell_P - \ell_{cP}) \cos \theta_P &= x_P \\ \ell_{R0} \sin(\theta_0 + \theta_{R0}) + \ell_{R1} \sin(\theta_0 + \theta_{R0} + \theta_{R1}) + \ell_{R2} \sin(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) - (\ell_P - \ell_{cP}) \sin \theta_P &= y_P \end{aligned} \right\} \quad (71)$$

To get the Pfaffian form of Eq. (70), differentiate Eq. (71) and rearrange terms.

The result is

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 & A_{16} & -1 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 & A_{26} & 0 & -1 \\ A_{31} & 0 & 0 & A_{34} & A_{35} & A_{36} & -1 & 0 \\ A_{41} & 0 & 0 & A_{44} & A_{45} & A_{46} & 0 & -1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_0 \\ \dot{\theta}_{L1} \\ \dot{\theta}_{L2} \\ \dot{\theta}_{R1} \\ \dot{\theta}_{R2} \\ \dot{\theta}_P \\ \dot{x}_P \\ \dot{y}_P \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (72)$$

The constant term,  $A_0$ , is a zero vector. The individual element in the constraints matrix are given by the following equations

$$A_{16} = -\ell_{c_p} \sin \theta_p \quad (73)$$

$$A_{26} = \ell_{c_p} \cos \theta_p \quad (74)$$

$$A_{36} = (\ell_p - \ell_{c_p}) \sin \theta_p \quad (75)$$

$$A_{46} = -(\ell_p - \ell_{c_p}) \cos \theta_p \quad (76)$$

$$A_{45} = \ell_{R2} \cos (\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) \quad (77)$$

$$A_{44} = A_{45} + \ell_{R1} \cos (\theta_0 + \theta_{R0} + \theta_{R1}) \quad (78)$$

$$A_{41} = A_{44} + \ell_{R0} \cos (\theta_0 + \theta_{R0}) \quad (79)$$

$$A_{35} = -\ell_{R2} \sin (\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) \quad (80)$$

$$A_{34} = A_{35} - \ell_{R1} \sin (\theta_0 + \theta_{R0} + \theta_{R1}) \quad (81)$$

$$A_{31} = A_{34} - \ell_{R0} \sin (\theta_0 + \theta_{R0}) \quad (82)$$

$$A_{23} = \ell_{L2} \cos (\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) \quad (83)$$

$$A_{22} = A_{23} + \ell_{L1} \cos (\theta_0 + \theta_{L0} + \theta_{L1}) \quad (84)$$

$$A_{21} = A_{22} + \ell_{L0} \cos (\theta_0 + \theta_{L0}) \quad (85)$$

$$A_{13} = -\ell_{L2} \sin (\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) \quad (86)$$

$$A_{12} = A_{13} - \ell_{L1} \sin (\theta_0 + \theta_{L0} + \theta_{L1}) \quad (87)$$

$$A_{11} = A_{12} - \ell_{L0} \sin (\theta_0 + \theta_{L0}) \quad (88)$$

If the manipulators are mounted on a fixed platform rather than a rotating base, an additional constraint equation is included in the A matrix. The constraint is that  $\theta_0$  is constant and therefore

$$\dot{\theta}_0 = 0 \quad (89)$$

This constraint is augmented into the A matrix by adding a fifth row. The first element in the row is a one. The remaining seven elements are all zeros.

### C. SIMPLIFIED EQUATIONS OF MOTION

The potential energy term is zero because motion is confined to the horizontal plane and the system is composed of rigid members. The inertia matrix, G matrix, B matrix, and constraints matrix can be found from the results of the previous sections. The remaining unknowns are the actuator torques and the Lagrange multipliers. By using the equations of motion and the Pfaffian form of the constraints, one can eliminate the Lagrange multipliers. The time derivative of Eq. (70) is

$$A\ddot{q} + \dot{A}\dot{q} = 0 \quad (90)$$

Solving Eq. (10) for  $\ddot{q}$  and substituting the result into Eq. (90) permits one to find an expression for the Lagrange multipliers.

$$\lambda = (AM^{-1}A^T)^{-1} (AM^{-1}(G - Bu) - A\ddot{q}) \quad (91)$$

The inertia matrix is always a square matrix with full rank and therefore invertible. To investigate the invertibility of  $AM^{-1}A^T$ , begin by creating a 4x4 matrix out of the third, fifth, seventh, and eighth columns of the constraints matrix.

$$\begin{bmatrix} A_{13} & 0 & -1 & 0 \\ A_{23} & 0 & 0 & -1 \\ 0 & A_{35} & -1 & 0 \\ 0 & A_{45} & 0 & -1 \end{bmatrix} \quad (92)$$

Inspection of this submatrix reveals that all of the rows and columns are linearly independent even if  $A_{13} = A_{23}$  and  $A_{35} = A_{45}$ . Therefore, the A matrix always has rank of 4. The 4x4 matrix product  $AM^{-1}A^T$  will also always have rank of 4 and is therefore invertible. Eq. (91) can be substituted back into the equations of motion (Eq. (10)) leaving the actuator torques as the only unknowns. The resulting equations of motion in which the Lagrange multipliers have been removed and potential energy is zero are

$$M\ddot{q} + \tilde{G} = \tilde{B}u \quad (93)$$

where

$$\tilde{G} = G - A^T (AM^{-1}A^T)^{-1} (AM^{-1}G - A\ddot{q}) \quad (94)$$

$$\tilde{B} = \left( I - A^T (AM^{-1}A^T)^{-1} AM^{-1} \right) B \quad (95)$$

#### D. REFERENCE TORQUES

Given a reference trajectory of the payload with known displacements, velocities and accelerations, one can use the simplified equations of motion (Eq. (93)) to solve for the actuator torques that will produce the reference trajectory.

$$\mathbf{M}_{\text{ref}}\ddot{\mathbf{q}}_{\text{ref}} + \tilde{\mathbf{G}}_{\text{ref}} = \tilde{\mathbf{B}}_{\text{ref}}\mathbf{u}_{\text{ref}} \quad (96)$$

The equations for specific elements in the matrices of Eq. (96) are the same as already presented. The subscript "ref" merely means that the displacement, velocity and acceleration terms are the values from the reference trajectory.

In this study, the total number of actuators is more than the system degrees of freedom. This situation is caused by the geometric constraints of multiple manipulators handling a common object. As a result, there are an infinity of solutions for the reference torques. One method to select a specific solution is to establish a cost function. An obvious cost function is to minimize a weighted norm of the actuator torques.

$$J = \frac{1}{2} \mathbf{u}_{\text{ref}}^T \mathbf{W} \mathbf{u}_{\text{ref}} \quad (97)$$

The problem now becomes one of minimizing the cost function (Eq. (97)) subject to the constraint that the reference equations of motion (Eq. (96)) are satisfied. Augmenting the cost function with the constraint by means of another Lagrange multiplier leads to

$$J = \frac{1}{2} \mathbf{u}_{\text{ref}}^T \mathbf{W} \mathbf{u}_{\text{ref}} + \gamma^T (\tilde{\mathbf{B}}_{\text{ref}}\mathbf{u}_{\text{ref}} - \mathbf{M}_{\text{ref}}\ddot{\mathbf{q}}_{\text{ref}} - \tilde{\mathbf{G}}_{\text{ref}}) \quad (98)$$

The minimum of the augmented cost function is found by taking the gradient of Eq. (98) with respect to the reference torques and with respect to the Lagrange multiplier. Each of the gradients is set to zero.

$$\nabla_{\mathbf{u}_{\text{ref}}} J = 0 = \mathbf{W} \mathbf{u}_{\text{ref}} + \tilde{\mathbf{B}}_{\text{ref}}^T \gamma \quad (99)$$

$$\nabla_{\gamma} J = 0 = \tilde{\mathbf{B}}_{\text{ref}}\mathbf{u}_{\text{ref}} - \mathbf{M}_{\text{ref}}\ddot{\mathbf{q}}_{\text{ref}} - \tilde{\mathbf{G}}_{\text{ref}} \quad (100)$$

Eqs. (99) and (100) are two equations in two unknowns ( $\gamma$ ,  $\mathbf{u}_{\text{ref}}$ ). Eliminating  $\gamma$  results in an expression for the reference actuator torques.

$$u_{ref} = W_u^{-1} \tilde{B}_{ref}^T \left( \tilde{B}_{ref} W_u^{-1} \tilde{B}_{ref}^T \right)^{\dagger} \left( M_{ref} \ddot{q}_{ref} + \tilde{G}_{ref} \right) \quad (101)$$

Although the matrix product  $\tilde{B}_{ref} W_u^{-1} \tilde{B}_{ref}^T$  is an 8x8 matrix, it is not invertible. A pseudo-inverse is needed because the system has only four degrees of freedom. Therefore, the matrix product  $\tilde{B}_{ref} W_u^{-1} \tilde{B}_{ref}^T$  is rank deficient and has a rank of four at most. This expression for reference actuator torques minimizes the augmented cost function (Eq. (98)) at each instant in time. Although the value for the reaction wheel torque is calculated, it is not minimized by this function. The reaction wheel torque profile is dictated by the disturbance torques transmitted to the centerbody as a result of manipulator and payload motion. For a given reference trajectory, an infinite variety of joint actuator torques can produce that trajectory. However, a given reference trajectory has only one reaction wheel torque profile that is common to all the infinity of joint torque combinations associated with that trajectory. Equation (101) selects from among the infinity of joint actuator torques the one combination that minimizes the weighted norm cost function. Although the selection is limited to a single choice, Equation (101) also produces the correct reaction wheel torque for the given reference trajectory.

### E. LYAPUNOV CONTROLLER

This material in this section is based on Ref. 16. The purpose of any control law is to provide system performance that satisfies a specification. As a bare minimum, the control law must keep the system stable. Because of the highly nonlinear nature of this spacecraft robotics system, most control laws simply do not apply. The motivation behind using Lyapunov methods is to develop a control law with guaranteed stability. Recall the equations of motion of the manipulator system are

$$M\ddot{q} + \tilde{G} = \tilde{B}u \quad (102)$$



Solving Eq. (102) for  $\ddot{\mathbf{q}}$  results in

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1} (\hat{\mathbf{B}}\mathbf{u} - \tilde{\mathbf{G}}) \quad (103)$$

Substituting Eqs. (94) and (95) back into Eq. (103) and grouping terms according to the form

$$\ddot{\mathbf{q}} = \mathbf{C}_1 \mathbf{u} + \mathbf{C}_2 \dot{\mathbf{q}} + \mathbf{C}_3 \quad (104)$$

leads to the following expressions

$$\mathbf{C}_1 = \mathbf{M}^{-1} \{ \mathbf{I} - \mathbf{A}^T (\mathbf{A} \mathbf{M}^{-1} \mathbf{A}^T)^{-1} \mathbf{A} \mathbf{M}^{-1} \} \mathbf{B} \quad (105)$$

$$\mathbf{C}_2 = -\mathbf{M}^{-1} \mathbf{A}^T (\mathbf{A} \mathbf{M}^{-1} \mathbf{A}^T)^{-1} \mathbf{A} \quad (106)$$

$$\mathbf{C}_3 = \mathbf{M}^{-1} \{ \mathbf{A}^T (\mathbf{A} \mathbf{M}^{-1} \mathbf{A}^T)^{-1} \mathbf{A} \mathbf{M}^{-1} - \mathbf{I} \} \mathbf{G} \quad (107)$$

Similarly, the reference maneuver accelerations can be expressed as

$$\ddot{\mathbf{q}}_{ref} = \mathbf{C}_{1_{ref}} \mathbf{u}_{ref} + \mathbf{C}_{2_{ref}} \dot{\mathbf{q}}_{ref} + \mathbf{C}_{3_{ref}} \quad (108)$$

where again the reference subscripts on the  $\mathbf{C}$  matrices indicate that reference maneuver values need to be used in their calculation. Let error quantities between the actual variables and their reference maneuver counterparts be defined by

$$\delta \mathbf{q} = \mathbf{q} - \mathbf{q}_{ref} \quad (109)$$

$$\delta \dot{\mathbf{q}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_{ref} \quad (110)$$

$$\delta \ddot{\mathbf{q}} = \ddot{\mathbf{q}} - \ddot{\mathbf{q}}_{ref} \quad (111)$$

Now define an error Lyapunov function as

$$U = 0.5 (\delta \dot{\mathbf{q}} \cdot \delta \dot{\mathbf{q}}) + f(\delta \mathbf{q}) \quad (112)$$

where  $f(\delta \mathbf{q}) \geq 0$ . Differentiating Eq. (112) results in

$$\dot{U} = \delta \dot{q} \cdot \delta \ddot{q} + \sum_i \frac{\partial f}{\partial (\delta q_i)} \delta \dot{q}_i \quad (113)$$

Let

$$\underline{F} = \left[ \frac{\partial f}{\partial (\delta q_1)} \quad \frac{\partial f}{\partial (\delta q_2)} \quad \cdots \quad \frac{\partial f}{\partial (\delta q_7)} \right]^T \quad (114)$$

Then Eq. (114) can be rewritten as

$$\dot{U} = \delta \dot{q} \cdot (\delta \ddot{q} + \underline{F}) \quad (115)$$

Substituting Eq. (104) and Eq. (108) into Eq. (111) and then Eq. (111) into Eq. (115) produces

$$\dot{U} = \delta \dot{q} \cdot \{ (C_1 \underline{u} - C_{1_{ref}} \underline{u}_{ref}) + (C_2 \dot{q} - C_{2_{ref}} \dot{q}_{ref}) + (C_3 - C_{3_{ref}}) + \underline{F} \} \quad (116)$$

If one lets the quantity inside the brackets of Eq. (116) equal  $-K_v \delta \dot{q}$  where  $K_v$  is a positive definite matrix, then one is guaranteed that  $\dot{U} \leq 0$  and therefore the system will be stable in the Lyapunov sense. Solving Eq. (116) for command torques,  $\underline{u}$ , leads to

$$\underline{u} = C_1^\dagger \{ -K_v \delta \dot{q} + C_{1_{ref}} \underline{u}_{ref} - (C_2 \dot{q} - C_{2_{ref}} \dot{q}_{ref}) - (C_3 - C_{3_{ref}}) - \underline{F} \} \quad (117)$$

$C_1$  is an  $8 \times 7$  matrix so  $C_1^\dagger$  is its pseudo inverse. Equation (119) finds the torques that should be used rather than the reference torques. All that remains is to choose a function for  $f(\delta q)$ . One can choose

$$f(\delta q) = \frac{1}{2} \delta q^T K_p \delta q \quad (118)$$

where like  $K_v$ ,  $K_p$  is required to be positive definite. Selection of values for the gain matrices is beyond the scope of this work. The simulations included in the next chapter use diagonal matrices with uniform values simply as a matter of convenience. One might try to adapt the linear quadratic regulator (LQR) problem to find more optimal gains.

After substituting Eq. (118) into Eq. (114) and that result into Eq. (119), one obtains the final form of the Lyapunov controller.

$$\underline{u} = C_1^{\dagger} \{ -K_v \delta \dot{q} + C_{1_{ref}} \underline{u}_{ref} - (C_2 \dot{q} - C_{2_{ref}} \dot{q}_{ref}) - (C_3 - C_{3_{ref}}) - K_p \delta q \} \quad (119)$$

If the differences between the reference trajectory and the system dynamics are small, the Lyapunov controller approaches the form of a proportional plus derivative (PD) control law.

## F. REFERENCE TRAJECTORIES

The reference trajectories describe the nominal path that the system follows in moving from the initial conditions to the desired final conditions. One need only specify reference trajectories for as many generalized coordinates as there are degrees of freedom. In effect, the generalized coordinates can be divided into two sets. One set contains the minimum number of coordinates needed to completely describe the system. The second set contains all remaining coordinates, (redundant coordinates). The choice of which generalized coordinates to specify is entirely arbitrary. A reasonable choice includes the payload coordinates and centerbody attitude since the user will probably be especially interested in these generalized coordinates. The redundant coordinates are the four manipulator joint angles. Given reference trajectories for the minimum number of coordinates exist, the redundant generalized coordinates can be derived. This research assumes trajectories are available which define displacement, velocity and acceleration for the centerbody attitude, payload attitude, and payload center of mass coordinates ( $X_P$ ,  $Y_P$ ,  $\theta_P$  and  $\theta_0$ ).

### 1. Calculating Redundant Coordinates

Figure 2 illustrates the relevant geometrical relationships to find the joint angles of the left manipulator.  $X_P$ ,  $Y_P$ ,  $\theta_P$  and  $\theta_0$  are obtained from the reference trajectory. Point LS is the left shoulder joint. It has Cartesian coordinates given by



The distance between the left shoulder and Point Q is given by

$$LSQ = \sqrt{(Q_x - LS_x)^2 + (Q_y - LS_y)^2} \quad (124)$$

The inertial angle formed by the vector from LS to Q is

$$\beta_1 = \text{atan} \left( \frac{Q_y - LS_y}{Q_x - LS_x} \right) \quad (125)$$

The dimensions of the triangle formed by the manipulator joints are known. Using the law of cosines, the interior angles at the shoulder and elbow can be found from

$$\beta_2 = \text{acos} \left( \frac{\ell_{L1}^2 + LSQ^2 - \ell_{L2}^2}{2\ell_{L1}LSQ} \right) \quad (126)$$

$$\beta_3 = \text{acos} \left( \frac{\ell_{L1}^2 + \ell_{L2}^2 - LSQ^2}{2\ell_{L1}\ell_{L2}} \right) \quad (127)$$

All that remains is to algebraically construct the manipulator joint angles from other angles as follows

$$\theta_{L1} = \beta_1 + \beta_2 - (\theta_0 + \theta_{L0}) \quad (128)$$

$$\theta_{L2} = \beta_3 + 180^\circ \quad (129)$$

The development for the right manipulator is similar. Its geometry is depicted in Figure 3.

Point RS is the right shoulder joint with Cartesian coordinate

$$RS_x = \ell_{R0} \cos (\theta_0 + \theta_{R0}) \quad (130)$$

$$RS_y = \ell_{R0} \sin (\theta_0 + \theta_{R0}) \quad (131)$$

Point P is the joint between the manipulator end and the payload. The Cartesian coordinates of this point are

$$P_x = X_P + (\ell_P - \ell_{cP}) \cos \theta_P \quad (132)$$

$$P_y = Y_P + (\ell_P - \ell_{cP}) \sin \theta_P \quad (133)$$

The distance between the right shoulder and Point P is given by



$$\beta_6 = \arccos \left( \frac{\ell_{R1}^2 + \ell_{R2}^2 - RSP^2}{2\ell_{R1}\ell_{R2}} \right) \quad (137)$$

The geometry in Figure 3 gives the manipulator joint angles based on the other angles.

$$\theta_{R1} = \beta_4 - \beta_5 - (\theta_0 + \theta_{R0}) \quad (138)$$

$$\theta_{R2} = 180^\circ - \beta_6 \quad (139)$$

Recall from the discussion of the Lyapunov controller that torques are calculated based not only on the generalized coordinates but their velocities and accelerations as well. The redundant coordinates have just been found, but the redundant coordinates velocities and accelerations must still be developed.

Differentiating Eqs. (122) and (123) expresses the velocity of Point Q.

$$\dot{Q}_x = \dot{X}_p + \dot{\theta}_p \ell_{cp} \sin \theta_p \quad (140)$$

$$\dot{Q}_y = \dot{Y}_p - \dot{\theta}_p \ell_{cp} \cos \theta_p \quad (141)$$

But the coordinates of Point Q can also be expressed in terms of left manipulator variables.

$$Q_x = \ell_{L0} \cos(\theta_0 + \theta_{L0}) + \ell_{L1} \cos(\theta_0 + \theta_{L0} + \theta_{L1}) + \ell_{L2} \cos(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) \quad (142)$$

$$Q_y = \ell_{L0} \sin(\theta_0 + \theta_{L0}) + \ell_{L1} \sin(\theta_0 + \theta_{L0} + \theta_{L1}) + \ell_{L2} \sin(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) \quad (143)$$

Differentiate these equations and rearrange the terms in the form of

$$\begin{bmatrix} \dot{Q}_x \\ \dot{Q}_y \end{bmatrix} = D_1 \dot{\theta}_0 + D_2 \begin{bmatrix} \dot{\theta}_{L1} \\ \dot{\theta}_{L2} \end{bmatrix} \quad (144)$$

where

$$D_2(1, 1) = -\ell_{L2} \sin(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) - \ell_{L1} \sin(\theta_0 + \theta_{L0} + \theta_{L1}) \quad (145)$$

$$D_2(1, 2) = -\ell_{L2} \sin(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) \quad (146)$$

$$D_2(2, 1) = \ell_{L2} \cos(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) + \ell_{L1} \cos(\theta_0 + \theta_{L0} + \theta_{L1}) \quad (147)$$

$$D_2(2,2) = \ell_{L2} \cos(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) \quad (148)$$

$$D_1(1,1) = -\ell_{L2} \sin(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) - \ell_{L1} \sin(\theta_0 + \theta_{L0} + \theta_{L1}) - \ell_{L0} \sin(\theta_0 + \theta_{L0}) \quad (149)$$

$$D_1(1,2) = \ell_{L2} \cos(\theta_0 + \theta_{L0} + \theta_{L1} + \theta_{L2}) + \ell_{L1} \cos(\theta_0 + \theta_{L0} + \theta_{L1}) + \ell_{L0} \cos(\theta_0 + \theta_{L0}) \quad (150)$$

Left manipulator joint velocities are found by rearranging Equation (144).

$$\begin{bmatrix} \dot{\theta}_{L1} \\ \dot{\theta}_{L2} \end{bmatrix} = D_2^{-1} \left( \begin{bmatrix} \dot{Q}_x \\ \dot{Q}_y \end{bmatrix} - D_1 \dot{\theta}_0 \right) \quad (151)$$

where Eqs. (140) and (141) provide the expressions for  $\dot{Q}_x$  and  $\dot{Q}_y$  respectively.

Using the same approach to find the joint velocities of the right manipulator, Point P is expressed as a function of right manipulator variables

$$P_x = \ell_{R0} \cos(\theta_0 + \theta_{R0}) + \ell_{R1} \cos(\theta_0 + \theta_{R0} + \theta_{R1}) + \ell_{R2} \cos(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) \quad (152)$$

$$P_y = \ell_{R0} \sin(\theta_0 + \theta_{R0}) + \ell_{R1} \sin(\theta_0 + \theta_{R0} + \theta_{R1}) + \ell_{R2} \sin(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) \quad (153)$$

Differentiate these equations and rearrange the terms in the form of

$$\begin{bmatrix} \dot{P}_x \\ \dot{P}_y \end{bmatrix} = D_3 \dot{\theta}_0 + D_4 \begin{bmatrix} \dot{\theta}_{R1} \\ \dot{\theta}_{R2} \end{bmatrix} \quad (154)$$

where

$$D_4(1,1) = -\ell_{R2} \sin(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) - \ell_{R1} \sin(\theta_0 + \theta_{R0} + \theta_{R1}) \quad (155)$$

$$D_4(1,2) = -\ell_{R2} \sin(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) \quad (156)$$

$$D_4(2,1) = \ell_{R2} \cos(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) + \ell_{R1} \cos(\theta_0 + \theta_{R0} + \theta_{R1}) \quad (157)$$

$$D_4(2,2) = \ell_{R2} \cos(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) \quad (158)$$

$$D_3(1,1) = -\ell_{R2} \sin(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) - \ell_{R1} \sin(\theta_0 + \theta_{R0} + \theta_{R1}) - \ell_{R0} \sin(\theta_0 + \theta_{R0}) \quad (159)$$

$$D_3(1,2) = \ell_{R2} \cos(\theta_0 + \theta_{R0} + \theta_{R1} + \theta_{R2}) + \ell_{R1} \cos(\theta_0 + \theta_{R0} + \theta_{R1}) + \ell_{R0} \cos(\theta_0 + \theta_{R0}) \quad (160)$$

Right manipulator joint velocities are found by rearranging Equation (154)



$$\begin{bmatrix} \dot{\theta}_{R1} \\ \dot{\theta}_{R2} \end{bmatrix} = D_4^{-1} \begin{bmatrix} \dot{P}_x \\ \dot{P}_y \end{bmatrix} - D_4 \dot{\theta}_0 \quad (161)$$

where expressions for  $\dot{P}_x$  and  $\dot{P}_y$  are found by differentiating Eqs. (132) and (133).

$$\dot{P}_x = \dot{X}_p - \dot{\theta}_p (\ell_p - \ell_{cp}) \sin \theta_p \quad (162)$$

$$\dot{P}_y = \dot{Y}_p + \dot{\theta}_p (\ell_p - \ell_{cp}) \cos \theta_p \quad (163)$$

Manipulator joint accelerations are found by differentiating the expressions for velocity (Eqs. (144) and (154)).

$$\begin{bmatrix} \ddot{Q}_x \\ \ddot{Q}_y \end{bmatrix} = \dot{D}_1 \dot{\theta}_0 + D_1 \ddot{\theta}_0 + D_2 \begin{bmatrix} \ddot{\theta}_{L1} \\ \ddot{\theta}_{L2} \end{bmatrix} + D_2 \begin{bmatrix} \dot{\theta}_{L1} \\ \dot{\theta}_{L2} \end{bmatrix} \quad (164)$$

$$\begin{bmatrix} \ddot{P}_x \\ \ddot{P}_y \end{bmatrix} = \dot{D}_3 \dot{\theta}_0 + D_3 \ddot{\theta}_0 + D_4 \begin{bmatrix} \ddot{\theta}_{R1} \\ \ddot{\theta}_{R2} \end{bmatrix} + D_4 \begin{bmatrix} \dot{\theta}_{R1} \\ \dot{\theta}_{R2} \end{bmatrix} \quad (165)$$

Solving for joint acceleration gives

$$\begin{bmatrix} \ddot{\theta}_{L1} \\ \ddot{\theta}_{L2} \end{bmatrix} = D_2^{-1} \left( \begin{bmatrix} \ddot{Q}_x \\ \ddot{Q}_y \end{bmatrix} - \dot{D}_1 \dot{\theta}_0 - D_1 \ddot{\theta}_0 - D_2 \begin{bmatrix} \dot{\theta}_{L1} \\ \dot{\theta}_{L2} \end{bmatrix} \right) \quad (166)$$

$$\begin{bmatrix} \ddot{\theta}_{R1} \\ \ddot{\theta}_{R2} \end{bmatrix} = D_4^{-1} \left( \begin{bmatrix} \ddot{P}_x \\ \ddot{P}_y \end{bmatrix} - \dot{D}_3 \dot{\theta}_0 - D_3 \ddot{\theta}_0 - D_4 \begin{bmatrix} \dot{\theta}_{R1} \\ \dot{\theta}_{R2} \end{bmatrix} \right) \quad (167)$$

where the accelerations of Points Q and P come from differentiating Eqs. (140)-(141) and Eqs. (162)-(163). Derivatives of the D matrices are constructed by differentiating Eqs. (145)-(150) and Eqs. (155)-(160).

## 2. Selecting Reference Trajectories

Any path which connects the associated endpoints can be a reference trajectory. To help ensure that the spacecraft and payload do not experience any unnecessary jerk or excitation of flexible structures, one might further constrain the path such that the veloci-

ties and accelerations are zero at the endpoints. Because a reaction wheel is required to maintain spacecraft attitude, the reaction wheel torque history is a prime candidate for optimization. Possible performance indices include the integral of the absolute value of reaction wheel torque

$$J = \int_{t_0}^{t_f} |u_{\text{wheel}}| dt \quad (168)$$

or the maximum reaction wheel torque.

$$J = \max(|u_{\text{wheel}}|) \quad (169)$$

A rigorous method for reference trajectory selection is to develop an optimal control solution to the two point boundary value problem. The performance index in the optimal control problem is given by

$$J = \int L[x(t), u(t), t] dt \quad (170)$$

Using Eq. (168) as an example,

$$L = |u_{\text{wh}}| = |Du| \quad (171)$$

where

$$u = [u_{\text{wh}} \ u_{\text{LS}} \ u_{\text{LE}} \ u_{\text{LW}} \ u_{\text{RS}} \ u_{\text{RE}} \ u_{\text{RW}}]^T \quad (172)$$

and

$$D = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \quad (173)$$

The state equations must be formulated as first order differential equations as

$$\dot{x} = f[x(t), u(t), t] \quad (174)$$

Because the system dynamics of my problem are second order differential equations, the state vector for the trajectory optimization is a combination of generalized displacements and velocities.

$$\mathbf{x} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{q} \end{bmatrix}^T \quad (175)$$

The resulting state equations are

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \ddot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{8 \times 8} & \mathbf{I}_{8 \times 8} \\ \mathbf{0}_{8 \times 8} & \mathbf{0}_{8 \times 8} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0}_{8 \times 7} \\ \mathbf{M}^{-1} \mathbf{B} \end{bmatrix} \mathbf{u} + \begin{bmatrix} \mathbf{0}_{8 \times 1} \\ -\mathbf{M}^{-1} \mathbf{G} \end{bmatrix} \quad (176)$$

where  $\tilde{\mathbf{G}}$  and  $\tilde{\mathbf{B}}$  are the same matrices as already found in Eqs. (94) and (95) respectively.

Desirable boundary conditions are such that the payload is at rest with zero acceleration at the beginning and end of the repositioning maneuver. However because the state vector does not contain accelerations, they cannot be specified as a boundary conditions. If the state vector is increased to include accelerations, then the first order state equations involve third order derivatives of the equations of motion rather than second order equations. This prevents including payload accelerations as part of the boundary conditions. To permit further development of the optimal control problem, the boundary conditions will be limited to desired positions and zero velocity.

$$\mathbf{x}(t_0) = \begin{bmatrix} \mathbf{q}(t_0) & \mathbf{0}_{1 \times 8} \end{bmatrix}^T \quad (177)$$

$$\mathbf{x}(t_f) = \begin{bmatrix} \mathbf{q}(t_f) & \mathbf{0}_{1 \times 8} \end{bmatrix}^T \quad (178)$$

The Hamiltonian formed by combining the performance index with the state equations is

$$H[\mathbf{x}(t), \mathbf{u}(t), \lambda(t), t] = L[\mathbf{x}(t), \mathbf{u}(t), t] + \lambda^T(t) \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] \quad (179)$$

The performance index and the state equations are both linear with respect to the control vector,  $\mathbf{u}$ . The consequences of this are that one cannot find a minimum by taking the gradient with respect to  $\mathbf{u}$  and setting it equal to zero. The applicable control form is bang-bang. Separating the Hamiltonian into those terms which premultiply  $\mathbf{u}$  and those which do not leads to the control law

$$\mathbf{u}^* = u_{\max} \text{sign}(\mathbf{H}_1) \quad (180)$$

where

$$\mathbf{H} = \mathbf{H}_{11} + \mathbf{H}_{12}\mathbf{u} \quad (181)$$

The other equations which must be satisfied are

$$\dot{\lambda}^T = -\frac{\partial \mathbf{H}}{\partial \mathbf{x}} = -\frac{\partial \mathbf{L}}{\partial \mathbf{x}} - \lambda^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \quad (182)$$

Because the performance index is only a function of  $\mathbf{u}$ , the first portion of the above necessary condition is trivial.

$$\frac{\partial \mathbf{L}}{\partial \mathbf{x}} = \mathbf{0}_{1 \times 16} \quad (183)$$

$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$  is not as easily found. The  $\mathbf{M}$ ,  $\mathbf{G}$ ,  $\mathbf{A}$ , and  $\mathbf{A}$  matrices are all functions of the state vector. In addition, the complexity is increased by several matrix inversions in the expression of  $\mathbf{f}$  in the  $\hat{\mathbf{G}}$  and  $\tilde{\mathbf{B}}$  matrices. Although an analytical expression may be theoretically possible, finding it was found to be extraordinarily tedious.

Recall, however, that the usefulness of the reference trajectories is to specify the generalized coordinates, velocities, and accelerations. Therefore, a convenient form for the reference trajectory is as a polynomial function of time. The following development uses the payload attitude generalized coordinate to illustrate how the polynomial reference trajectories are applied. Let

$$\Delta\theta_P = \theta_P(t_f) - \theta_P(t_0) \quad (184)$$

where  $t_0$  is the maneuver start time and  $t_f$  is the final time. The duration of the maneuver is the difference between  $t_f$  and  $t_0$ .  $\theta_P(t_0)$  and  $\theta_P(t_f)$  are the initial payload attitude and the desired final attitude respectively. If the desired reference path for the payload attitude in moving from initial to final conditions is a curve which can be represented as a polynomial function,  $f(\tau)$ , where  $\tau$  is simply normalized time

$$\tau = \frac{(t - t_0)}{(t_f - t_0)} \quad (185)$$

then

$$\theta_{p_{ref}}(t) = \theta_p(t_0) + f(\tau) (\Delta\theta_p) \quad (186)$$

$$\dot{\theta}_{p_{ref}}(t) = f'(\tau) (\Delta\theta_p) \left( \frac{1}{t_f - t_0} \right) \quad (187)$$

$$\ddot{\theta}_{p_{ref}}(t) = f''(\tau) (\Delta\theta_p) \left( \frac{1}{(t_f - t_0)^2} \right) \quad (188)$$

In order for Eq. (186) to produce the correct initial and final values for  $\theta_{p_{ref}}$ , the polynomial must be such that

$$f(\tau=0) = 0 \quad (189)$$

$$f(\tau=1) = 1 \quad (190)$$

To produce zero velocity and acceleration at the initial and final conditions requires that  $f(\tau)$  also satisfy

$$f'(\tau=0) = 0 \quad (191)$$

$$f'(\tau=1) = 0 \quad (192)$$

$$f''(\tau=0) = 0 \quad (193)$$

$$f''(\tau=1) = 0 \quad (194)$$

The minimum order polynomial which satisfies the boundary conditions of Eqs. (189)-(194) is

$$f(\tau) = 6\tau^5 - 15\tau^4 + 10\tau^3 \quad (195)$$

The expressions for payload reference trajectory using the fifth order polynomial become

$$\theta_{p_{ref}}(t) = \theta_p(t_0) + (6\tau^5 - 15\tau^4 + 10\tau^3) (\Delta\theta_p) \quad (196)$$

$$\theta_{p_{ref}}(t) = (30\tau^4 - 60\tau^3 + 30\tau^2) (\Delta\theta_p) \left( \frac{1}{t_f - t_0} \right) \quad (197)$$

$$\theta_{p_{ref}}(t) = (120\tau^3 - 180\tau^2 + 60\tau) (\Delta\theta_p) \left( \frac{1}{(t_f - t_0)^2} \right) \quad (198)$$

The polynomial reference trajectory is also be applied to the other generalized coordinates which form the minimum set to describe the system (i.e. centerbody attitude and payload center of mass coordinates). The redundant generalized coordinates are calculated from the reference coordinates as described earlier.

Higher order polynomials can increase the complexity of the path but offer the advantage that an infinity of polynomial coefficients satisfy the position, velocity, and acceleration boundary conditions. The selection of the coefficients affords an opportunity to optimize the reaction wheel torque. In this system, manipulator actuator torques are internal while the reaction wheel torque is the only external torque. Therefore, the reaction wheel torque will be equal to the rate of change of angular momentum which can be calculated directly from a reference trajectory. This technique is more computationally efficient because it does not require the construction of the  $\tilde{G}$  and  $\tilde{B}$  matrices.

In general, the angular momentum about the inertial origin for each member of the system is

$$\underline{H}_i = I_i \underline{\omega}_i + m_i (\underline{r}_i \times \underline{v}_i) \quad (199)$$

where  $I_i$  is the moment of inertia of the  $i^{\text{th}}$  body about its center of mass

$\underline{\omega}_i$  is the angular rate of the  $i^{\text{th}}$  body

$m_i$  is the mass of the  $i^{\text{th}}$  body

$\underline{r}_i$  is the inertial position of the  $i^{\text{th}}$  body center of mass

$\underline{v}_i$  is the inertial velocity of the  $i^{\text{th}}$  body center of mass

The angular rate, position and velocity vectors were previously developed in connection with determining kinetic energy. Those expressions require some coordinate transformations to express all the terms with respect to the inertial coordinate frame. The change in angular momentum is found by differentiating Eq. (199) to produce

$$\dot{H}_i = I_i \dot{\omega}_i + m_i (r_i \times a_i) \quad (200)$$

The total system change in angular momentum is the sum of change in angular momentum for each of the members. After collecting terms with common angular velocity or acceleration terms, the expression for the system change in angular momentum is given by

$$\ddot{H} = I_p \ddot{\theta}_p + m_p (\dot{X}_p \ddot{Y}_p - \ddot{X}_p \dot{Y}_p) \quad (201)$$

$$\begin{aligned} & + \ddot{\theta}_{L1} [I_{L1} + I_{L2} + m_{L1} l_{cL1}^2 + m_{L2} l_{cL2}^2 + m_{L2} l_{L1}^2 + m_{L1} l_{L0} l_{cL1} \cos \theta_{L1} + m_{L2} l_{L0} l_{L1} \cos \theta_{L1} \\ & \quad + 2m_{L2} l_{L1} l_{cL2} \cos \theta_{L2} + m_{L2} l_{L0} l_{cL2} \cos (\theta_{L1} + \theta_{L2})] \\ & + \ddot{\theta}_{L2} [I_{L2} + m_{L2} l_{cL2}^2 + m_{L2} l_{L1} l_{cL2} \cos \theta_{L2} + m_{L2} l_{L0} l_{cL2} \cos (\theta_{L1} + \theta_{L2})] \\ & + \ddot{\theta}_{R1} [I_{R1} + I_{R2} + m_{R1} l_{cR1}^2 + m_{R2} l_{cR2}^2 + m_{R2} l_{R1}^2 + m_{R1} l_{R0} l_{cR1} \cos \theta_{R1} + m_{R2} l_{R0} l_{R1} \cos \theta_{R1} \\ & \quad + 2m_{R2} l_{R1} l_{cR2} \cos \theta_{R2} + m_{R2} l_{R0} l_{cR2} \cos (\theta_{R1} + \theta_{R2})] \\ & + \ddot{\theta}_{R2} [I_{R2} + m_{R2} l_{cR2}^2 + m_{R2} l_{R1} l_{cR2} \cos \theta_{R2} + m_{R2} l_{R0} l_{cR2} \cos (\theta_{R1} + \theta_{R2})] \\ & + \ddot{\theta}_{L1} \ddot{\theta}_{L2} [-2m_{L2} l_{L1} l_{cL2} \sin \theta_{L2} - 2m_{L2} l_{L0} l_{cL2} \sin (\theta_{L1} + \theta_{L2})] \\ & + \ddot{\theta}_{L1}^2 [-m_{L1} l_{L0} l_{cL1} \sin \theta_{L1} - m_{L2} l_{L0} l_{L1} \sin \theta_{L1} - m_{L2} l_{L0} l_{cL2} \sin (\theta_{L1} + \theta_{L2})] \\ & + \ddot{\theta}_{L2}^2 [-m_{L2} l_{L1} l_{cL2} \sin \theta_{L2} - m_{L2} l_{L0} l_{cL2} \sin (\theta_{L1} + \theta_{L2})] \\ & + \ddot{\theta}_{R1} \ddot{\theta}_{R2} [-2m_{R2} l_{R1} l_{cR2} \sin \theta_{R2} - 2m_{R2} l_{R0} l_{cR2} \sin (\theta_{R1} + \theta_{R2})] \\ & + \ddot{\theta}_{R1}^2 [-m_{R1} l_{R0} l_{cR1} \sin \theta_{R1} - m_{R2} l_{R0} l_{R1} \sin \theta_{R1} - m_{R2} l_{R0} l_{cR2} \sin (\theta_{R1} + \theta_{R2})] \\ & + \ddot{\theta}_{R2}^2 [-m_{R2} l_{R1} l_{cR2} \sin \theta_{R2} - m_{R2} l_{R0} l_{cR2} \sin (\theta_{R1} + \theta_{R2})] \end{aligned}$$

Any polynomial reference trajectory that satisfies the initial condition concerning displacement cannot have a constant term. Polynomials which satisfy the velocity and acceleration initial conditions must not contain linear or quadratic terms. The general  $n^{\text{th}}$  order polynomial reference trajectory has the form

$$f(\tau) = a_n \tau^n + a_{n-1} \tau^{n-1} + a_{n-2} \tau^{n-2} + \dots + a_5 \tau^5 + a_4 \tau^4 + a_3 \tau^3 \quad (202)$$

Derivatives are

$$f'(\tau) = n a_n \tau^{n-1} + (n-1) a_{n-1} \tau^{n-2} + (n-2) a_{n-2} \tau^{n-3} + \dots + 5 a_5 \tau^4 + 4 a_4 \tau^3 + 3 a_3 \tau^2 \quad (203)$$



$$f'''(\tau) = n(n-1)a_n\tau^{n-2} + (n-1)(n-2)a_{n-1}\tau^{n-3} + (n-2)(n-3)a_{n-2}\tau^{n-4} + \dots + 20a_5\tau^5 + 12a_4\tau^4 + 6a_3\tau \quad (204)$$

When  $\tau=1$  and the final conditions,  $f(1) = 1$ ,  $f'(1) = 0$  nad  $f''(1) = 0$ , are substituted into Eqs. (202)-(204), these equations can be put into matrix form

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ n & (n-1) & (n-2) & \dots & 5 & 4 & 3 \\ n(n-1) & (n-1)(n-2) & (n-2)(n-3) & \dots & 20 & 12 & 6 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ \dots \\ a_5 \\ a_4 \\ a_3 \end{bmatrix} = [W]a \quad (205)$$

The column vector of polynomial coefficients can be partitioned. One segment,  $a_{543}$ , contains the coefficients for the third, fourth, and fifth order terms in Eq. (202). The other segment,  $a_{high}$ , contains all of the coefficients of order six and higher.

$$a = \begin{bmatrix} a_{high} \\ a_{543} \end{bmatrix} \quad (206)$$

The W matrix can be partitioned accordingly.

$$W = \begin{bmatrix} W_{high} & W_{543} \end{bmatrix} \quad (207)$$

One can then solve for the lower order polynomial coefficients in terms of the higher order coefficients by substituting Eqs. (206) and (207) into Eq. (205). The result specifies polynomial reference trajectory coefficients which satisfy the boundary conditions.

$$a_{543} = W_{543}^{-1} \left( \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - W_{high} a_{high} \right) \quad (208)$$

An optimal solution for a polynomial reference trajectory is found by using the MATLAB function `fminu`. This tool numerically finds the solution to an unconstrained function minimization problem using a quasi-Newton method. The function to be mini-

mized is the rate of change of angular momentum, Eq. (201), which can be found once a reference trajectory is specified. The user makes an initial guess for the higher order reference trajectory coefficients. The lower order coefficients are calculated by Eq. (208). The MATLAB function then varies the higher order coefficients and recalculates the lower order coefficients as necessary to minimize change in angular momentum. One limitation to this technique is that the algorithm may converge to a local rather than the global minimum.

### III. VALIDATION AND SIMULATION RESULTS

The computer simulations presented in this chapter were obtained using the MATLAB subroutines included in Appendix B. The integrator uses 4<sup>th</sup> and 5<sup>th</sup> order Runge-Kutta formulas. See Appendix B for documented listings of the computer code.

#### A. VALIDATION

To verify the equations and find the programming bugs, test cases were developed. The simulations are analyzed to ensure that universal principles such as conservation of energy and angular momentum are not violated. Numeric values for the generic dual two-link manipulator system are contained in Table 1. The generic model is the strawman configuration that all of the test cases are based on with the exception of a few minor variations. The variations will be pointed out in the appropriate test cases. The values for the generic model's system properties are picked for uniformity and simplicity. The manipulator links and the payload are modelled as slender rods of uniform density.

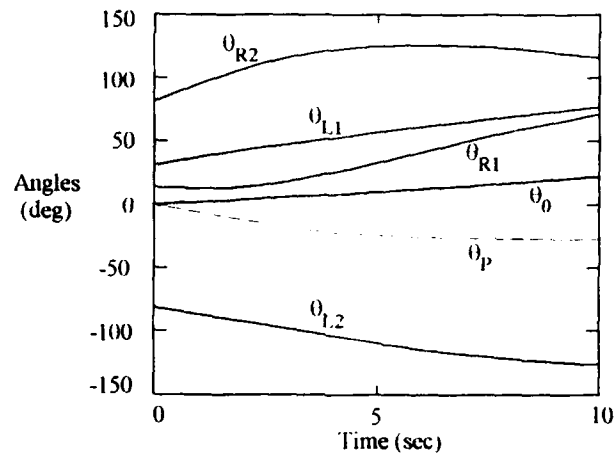
##### 1. Conservation of Kinetic Energy

In the first test case, no torques are applied and the initial velocities are nonzero. Under these conditions, the system links drift subject to the constraints of being pinned together. Since potential energy is zero and there are no external energy sources, kinetic energy should remain constant. The system begins with the payload parallel to a line drawn between the two shoulders and 0.75m away from them. The initial angular rate for the centerbody is chosen to be  $\dot{\theta}_0 = 2$  deg/sec. The initial angular rate for the payload is  $\dot{\theta}_p = -5$  deg/sec. Initial velocities for the payload center of mass are -0.1 m/sec along the x axis and -0.05 m/sec along the y axis. The remaining generalized velocities are calculated

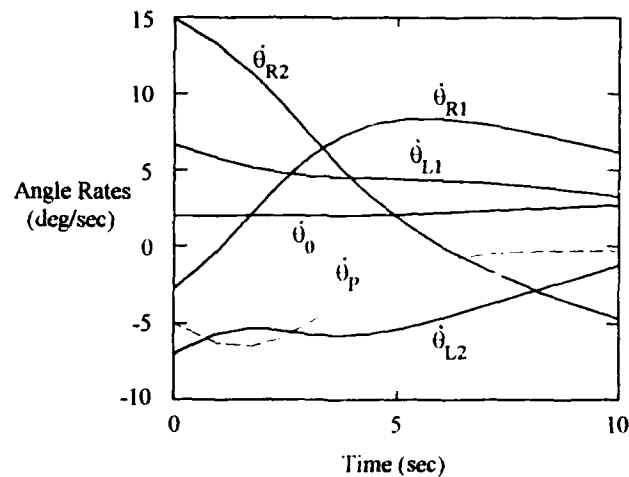
**TABLE 1. GENERIC MODEL SYSTEM PROPERTIES**

	Parameter	Value
Length (m)	$l_{L0}$	0.75
	$l_{L1}$	0.5
	$l_{L2}$	0.5
	$l_{R0}$	0.5
	$l_{R1}$	0.5
	$l_{R2}$	0.5
	$l_p$	$0.75\sqrt{2}$
Mass (kg)	$m_0$	5
	$m_{L1}$	1
	$m_{L2}$	1
	$m_{R1}$	1
	$m_{R2}$	1
	$m_p$	1
Center of Mass (m)	$l_{c0}$	0
	$l_{cL1}$	0.25
	$l_{cL2}$	0.25
	$l_{cR1}$	0.25
	$l_{cR2}$	0.25
	$l_{cp}$	0.25
Moments of Inertia (kg-m <sup>2</sup> )	$I_0$	5
	$I_{L1}$	0.02083
	$I_{L2}$	0.02083
	$I_{R1}$	0.02083
	$I_{R2}$	0.02083
	$I_p$	0.02083
Shoulder Location (deg)	$\theta_{L0}$	90
	$\theta_{R0}$	45

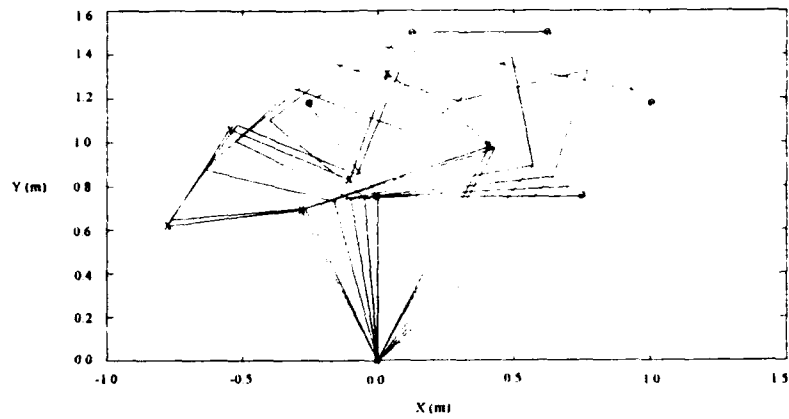
based on the values specified for the centerbody and payload. Initial angular rates for the manipulator links are  $\dot{\theta}_{L1} = 6.6607$  deg/sec,  $\dot{\theta}_{L2} = -7.0457$  deg/sec,  $\dot{\theta}_{R1} = -2.7553$  deg/sec, and  $\dot{\theta}_{R2} = 14.9127$  deg/sec. The graphical results from this test case are included in Figures 4-8. As indicated in Figure 7, kinetic energy is conserved in this case.



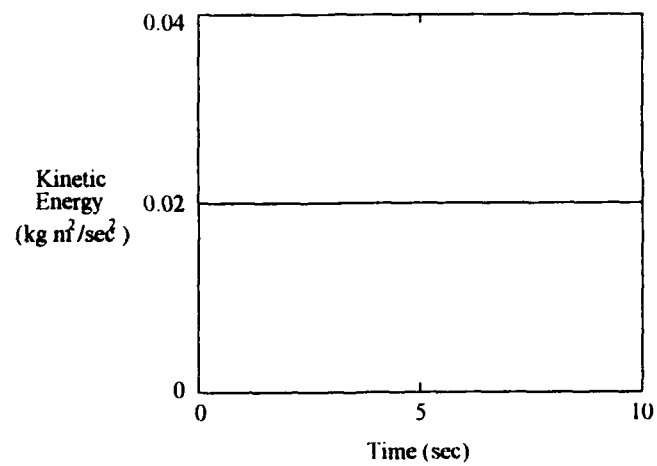
**Figure 4: Test Case 1 Angles**



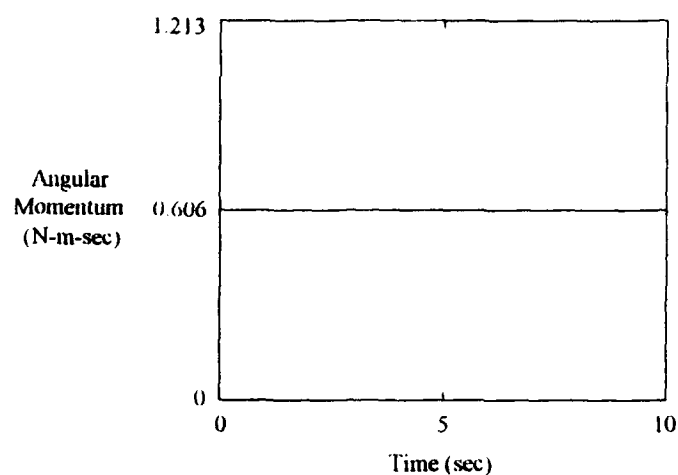
**Figure 5: Test Case 1 Angular Rates**



**Figure 6: Test Case 1 Time Lapse Stick Figure**



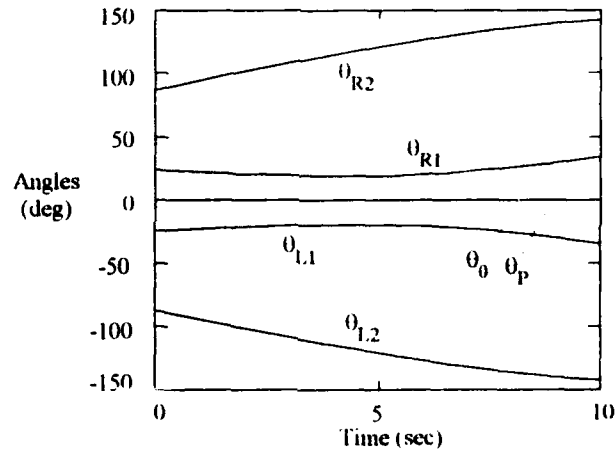
**Figure 7: Test Case 1 Kinetic Energy**



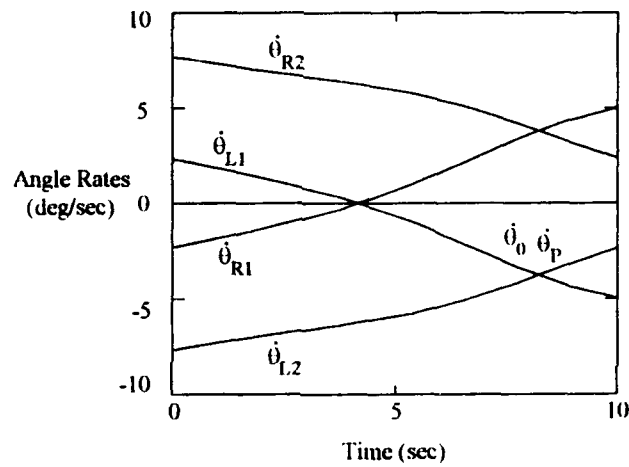
**Figure 8: Test Case 1 Angular Momentum**

Test Case 2 is an extension of Test Case 1. This is still a case with nonzero initial velocities and no external torques. However, the system geometry is altered to be symmetrical. In addition to conservation of kinetic energy, this test case will ensure that the symmetry is preserved. The physical alterations in the system involve moving the location of the left shoulder from 90 degrees to 135 degrees and decreasing the distance from the origin to the right shoulder to 0.75 meters. The payload still begins centered between the shoulders and parallel to the y axis but is 1.2 m from the origin. To maintain symmetry, the initial velocities must also be symmetrical. The initial angular rate for the centerbody is chosen to be  $\dot{\theta}_0 = 0$  deg/sec. The initial angular rate for the payload is also zero. Initial velocities for the payload center of mass are zero along the x axis and -0.05 m/sec along the y axis. The remaining generalized velocities are again calculated based on the values specified for the centerbody and payload. Initial angular rates for the manipulator links are  $\dot{\theta}_{L1} = 2.3188$  deg/sec,  $\dot{\theta}_{L2} = -7.6851$  deg/sec,  $\dot{\theta}_{R1} = -2.3188$  deg/sec, and  $\dot{\theta}_{R2} = 7.6851$  deg/sec. This combination of system geometry and initial velocities is designed to

cause the payload to drift toward the origin without changing its attitude. Figures 9-13 show the results from this test case. Kinetic energy is conserved and symmetry is preserved.

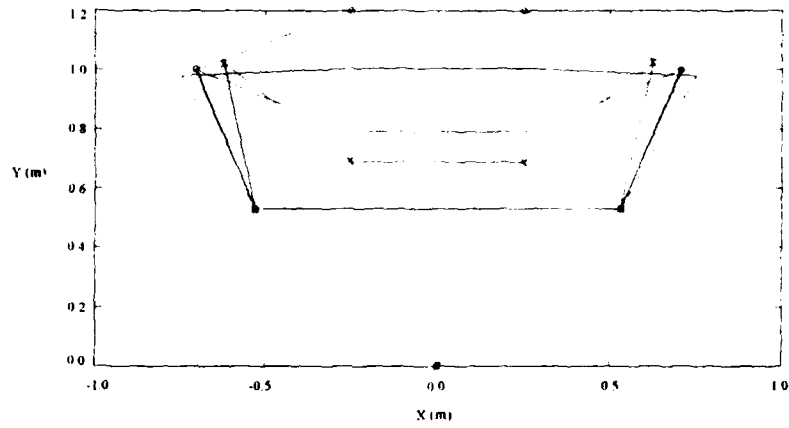


**Figure 9: Test Case 2 Angles**

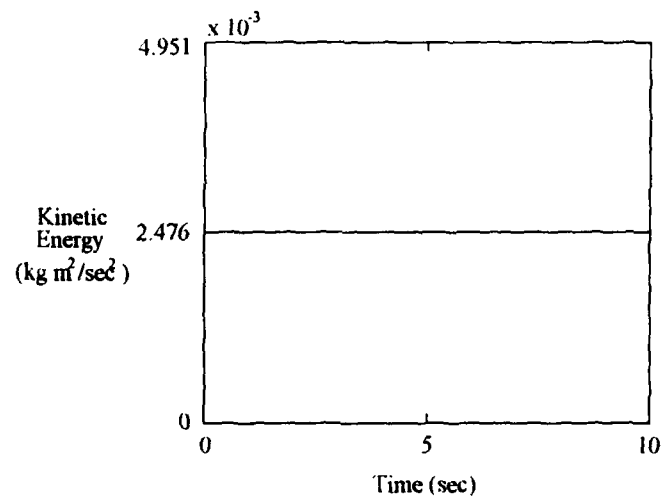


**Figure 10: Test Case 2 Angular Rates**

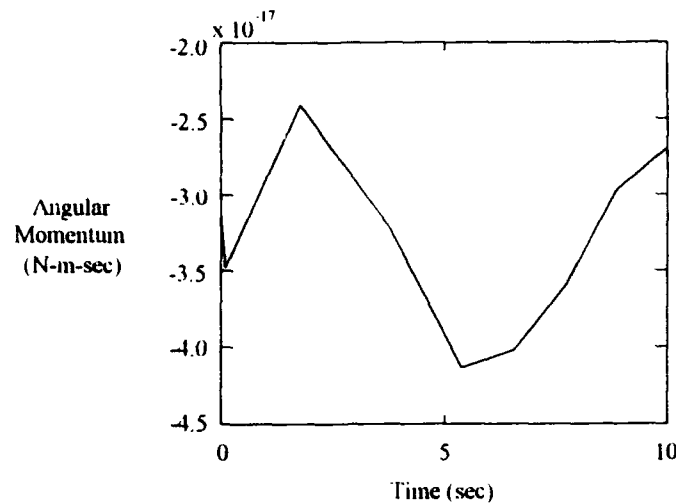




**Figure 11: Test Case 2 Time Lapse Stick Figure**



**Figure 12: Test Case 2 Kinetic Energy**



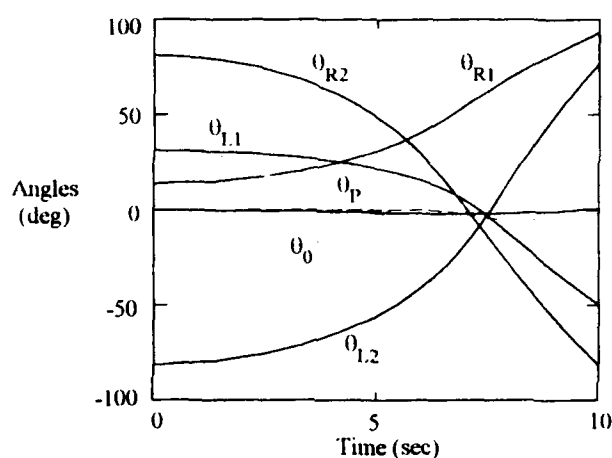
**Figure 13: Test Case 2 Angular Momentum**

## 2. Conservation of Angular Momentum

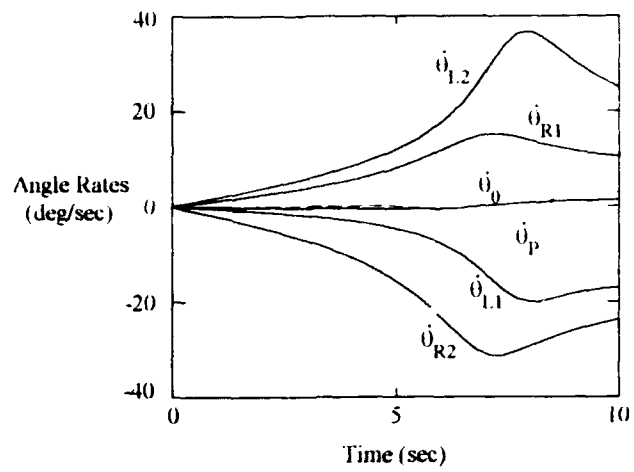
As long as a system does not include external torques, one expects that angular momentum should be conserved. The joint actuators provide internal torques while the reaction wheel is the only external source. Test Cases 1 and 2 did not include a reaction wheel and are therefore subject to investigation with respect to conservation of angular momentum. Both cases do satisfy the requirement as indicated by Figures 8 and 13. Furthermore, due to the symmetry in the system in Test Case 2, the angular momentum of the left manipulator links should be cancelled out by the angular momentum of the right manipulator links. The centerbody and payload should not have any angular momentum. Consequently, angular momentum for the system should not only be conserved, it should be zero. Figure 13 show that the angular momentum remained virtually zero. The non-zero values of about  $3 \times 10^{-17}$  are well within the integration algorithm tolerance of  $10^{-6}$ .

Test Case 3 returns to the generic system from Table 1. Initially, the system is at rest. Constant torques are applied at both shoulders and nowhere else. The torques are

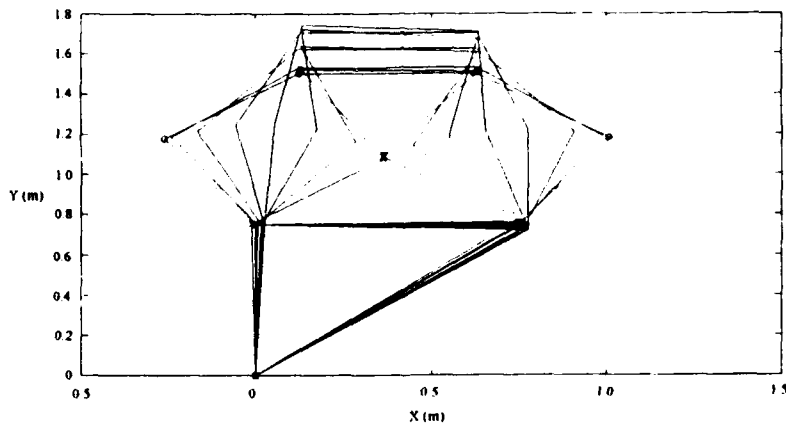
0.01 N-m applied in the positive direction at the right shoulder and the negative direction at the left shoulder. Because the joint torques are internal to the system, angular momentum must still be conserved even though kinetic energy won't be. Furthermore, since the system began at rest, the angular momentum should remain at zero. The results are shown in Figures 14-18. Although the angular momentum did not remain identically equal to zero, their magnitudes of less than  $2 \times 10^{-7}$  are within the  $10^{-6}$  tolerance placed on the integration algorithm.



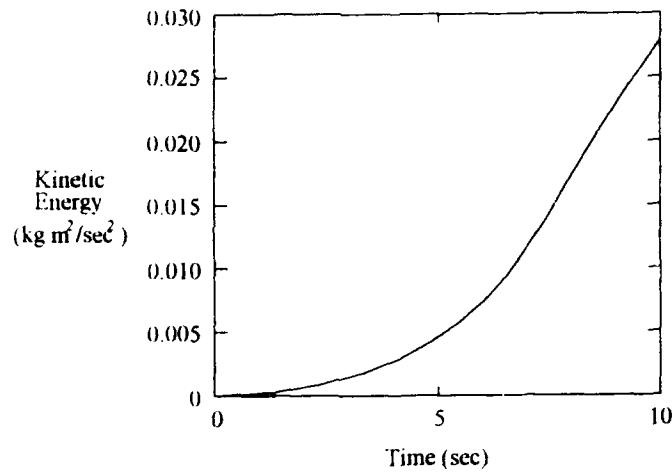
**Figure 14: Test Case 3 Angles**



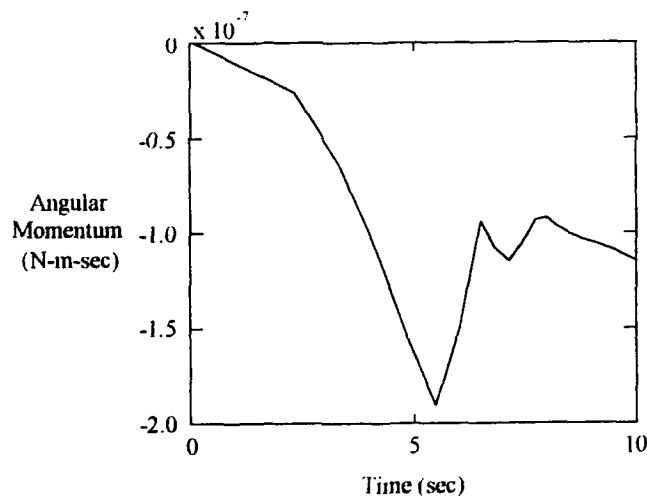
**Figure 15: Test Case 3 Angular Rates**



**Figure 16: Test Case 3 Time Lapse Stick Figure**



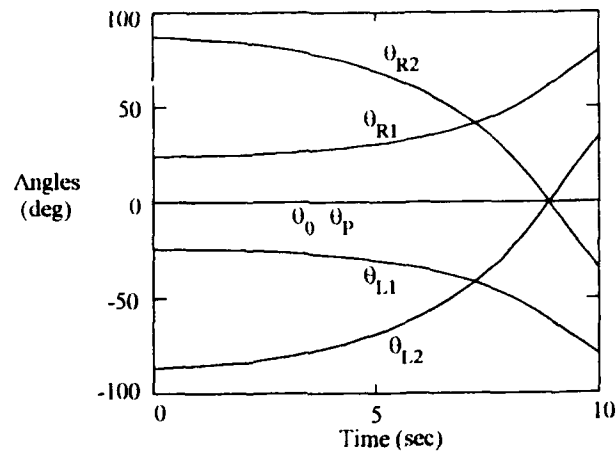
**Figure 17: Test Case 3 Kinetic Energy**



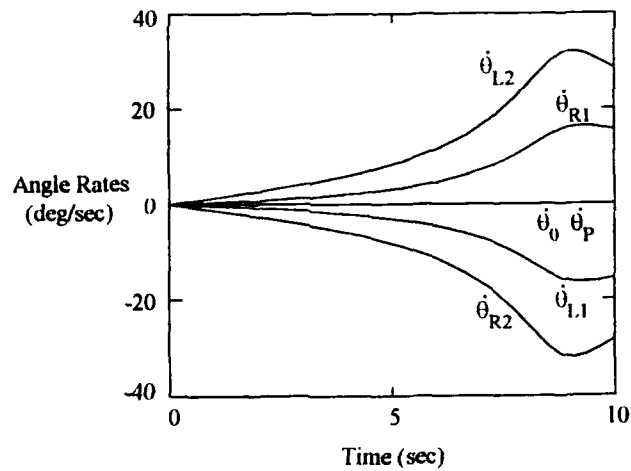
**Figure 18: Test Case 3 Angular Momentum**

Test Case 4 is similar to Test Case 3 but the symmetrical system geometry is used instead of the generic geometry. This change should produce symmetric motion and zero angular momentum. The reaction wheel is still disabled. Figures 19-23 indicate the sys-

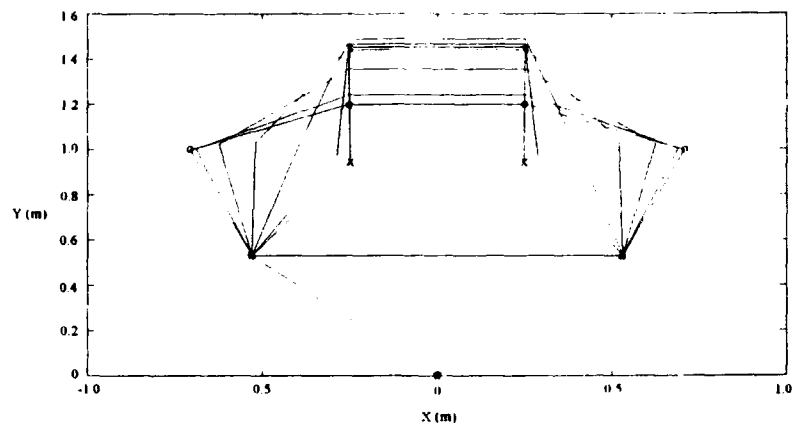
tem reacted as expected. Changing the torques to time varying profiles rather than constants led to similar results.



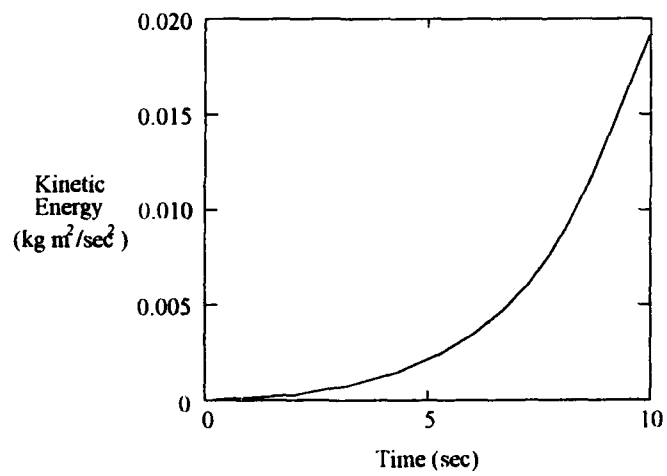
**Figure 19: Test Case 4 Angles**



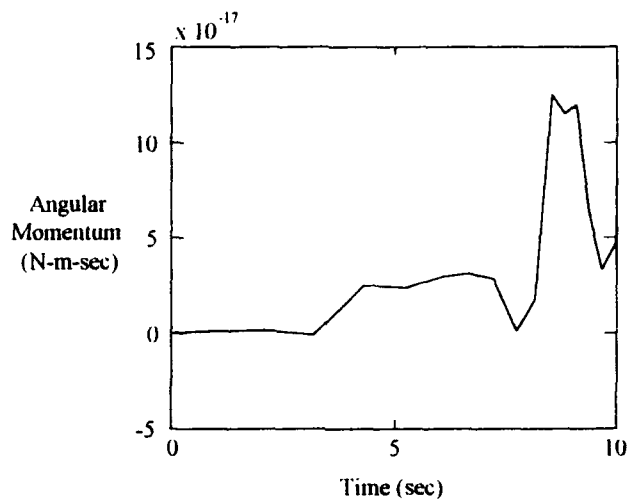
**Figure 20: Test Case 4 Angular Rates**



**Figure 21: Test Case 4 Time Lapse Stick Figure**



**Figure 22: Test Case 4 Kinetic Energy**



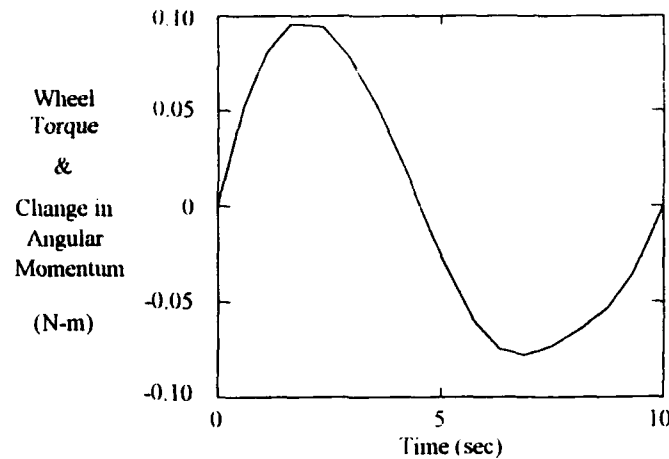
**Figure 23: Test Case 4 Angular Momentum**

### 3. Wheel Torque and Constraints

The remaining test cases involved using the reaction wheel on the centerbody. The wheel's function was to maintain attitude pointing. The system begins at rest. The torque applied by the wheel is an external torque in this model. Therefore, its value must be the same as the change in angular momentum. The wheel torque is found by means of the inverse kinematics equations in Chapter II. These calculations are entirely independent of finding the change in angular momentum. After a simulation is finished, a separate section in the program code calculates the change in angular momentum using the generalized coordinates, velocities and accelerations produced by the integration. These values are plotted along with those of the reaction wheel torque. A sample plot is contained in Figure 24. This particular plot is for the case of a fifth order polynomial reference trajectory. The rest of the plots associated with this case are presented later in the Simulations section. The validation tests concerning conservation of kinetic energy and angular momentum required special circumstances to create those conditions. The requirement



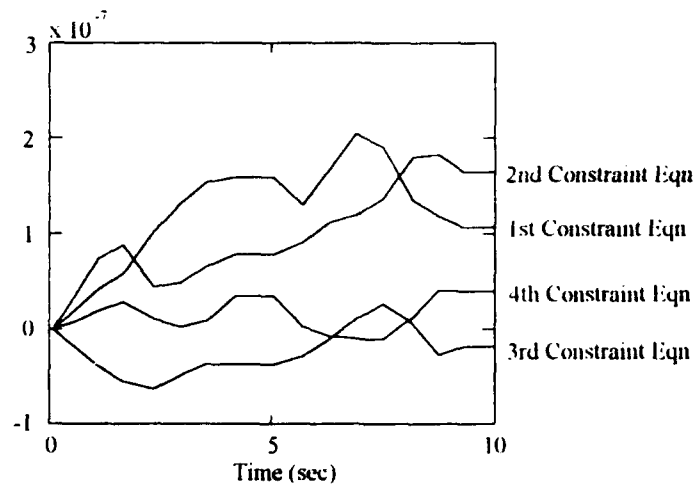
that the reaction wheel torque equal the change in angular momentum is more universal. It is a verification check performed with every simulation involving a reaction wheel.



**Figure 24: Sample of Wheel Torque and Change in Angular Momentum vs. Time**

An even more universal check also performed with every simulation is the requirement that the constraint equations ( $A\dot{q} + A_0 = 0$ ) are satisfied. Figure 25 shows a sample plot. This plot was also taken from the fifth order polynomial reference trajectory case. The values plotted represent the four constraint equations contained in Eqn 72. The non-zero values are attributed to numerical errors created by the integration.

Finally, a common sense check also performed with every simulation is simply to verify that the payload was repositioned to the desired final location. This cannot happen if the torques applied to the system were incorrect. This test is a necessary but not sufficient condition that the code operates correctly.



**Figure 25: Sample of Constraints vs. Time**

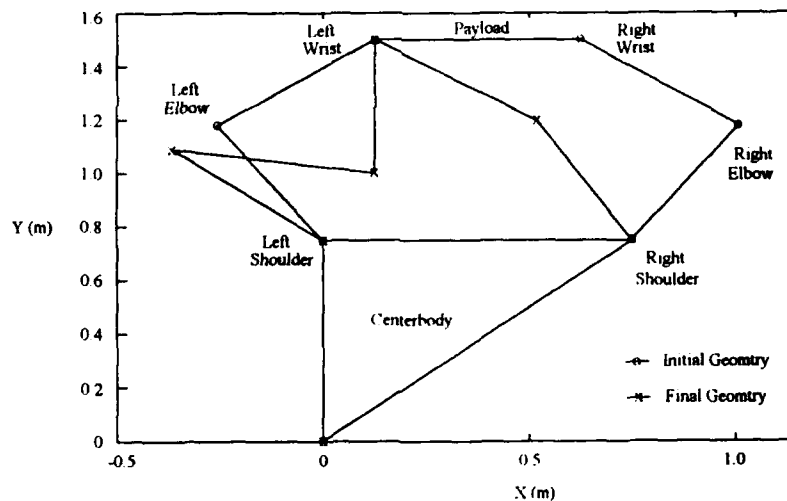
## B. SIMULATIONS

This section presents results from several simulations of an analytical model. The desired payload repositioning maneuver is illustrated in Figure 26. The final position for the payload involves a 90 degree rotation and the right endpoint finishes where the left endpoint started.

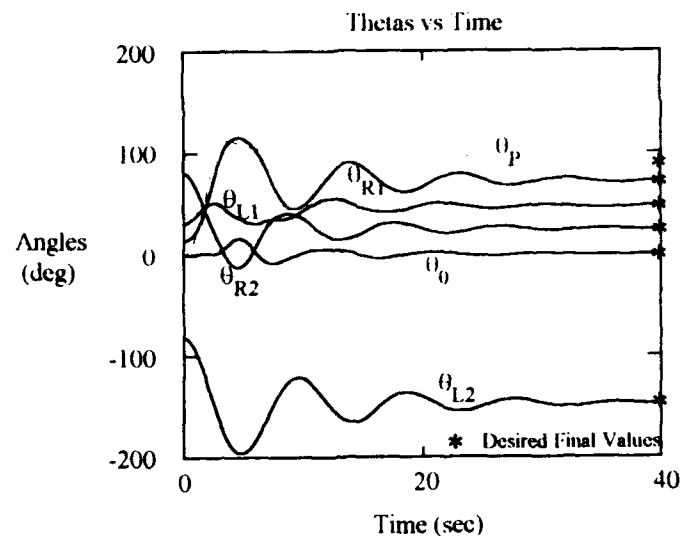
### 1. Lyapunov Point Controller

In the first simulation, the repositioning is done entirely by the Lyapunov controller without the benefit of a reference trajectory. The behavior is that of a point controller with an initial displacement rather than that of a tracking controller. Due to the absence of the weighted norm reference torques, this controller cannot be considered to have cooperative nature. Figure 27 presents the angular displacement history. The asterisks on the right side of the plot indicate the desired final angles. Although the system is approaching the desired final geometry, it has not completely settled down even after 40 seconds. Position errors (Figure 28) are still present as well as nonzero velocities (Figure 29). Also note that the reaction wheel torque is quite high during the maneuver (Figure 30). The joint

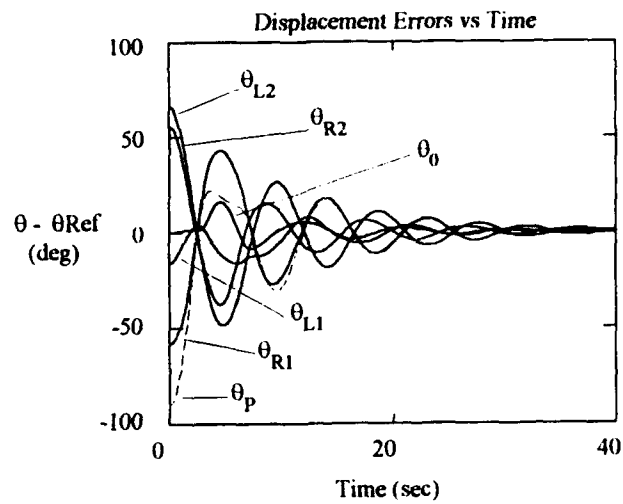
actuator torques are considerably less than the reaction wheel torque. They are not identified individually because the most important feature of Figure 30 is the reaction wheel torque. As a quantitative measure of this controller's quality,  $\int |u_{wh}| dt$  produces a value of 17.3841. The oscillatory nature of the system is evident in the angular position and velocity plots. Despite the oscillations, however, the stability of the controller is also illustrated. Figure 31 depicts the system geometry at several instances during the maneuver. The left manipulator links actually cross over each other. In experimental hardware, the links would collide instead. Figure 32 removes the clutter that is present in Figure 31 and displays only the initial and final geometry. The Lyapunov point controller also does a poor job of maintaining the centerbody attitude. This is clearly evident in Figures 27 and 31. The attitude error peaks at about 16 degrees.



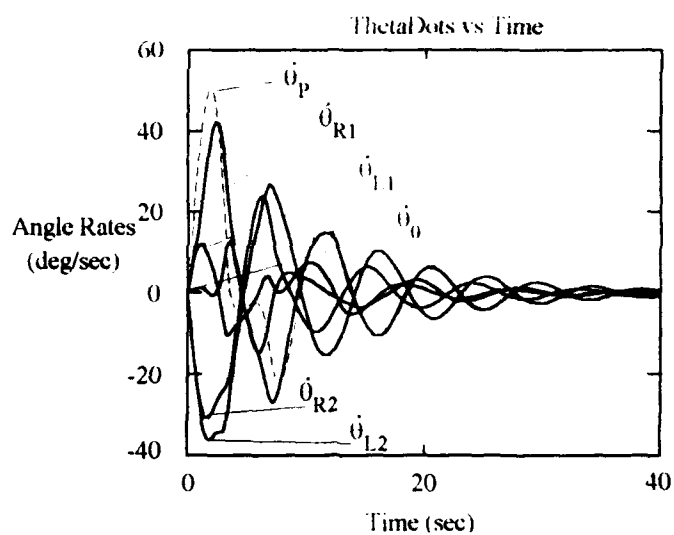
**Figure 26: Desired Repositioning Maneuver**



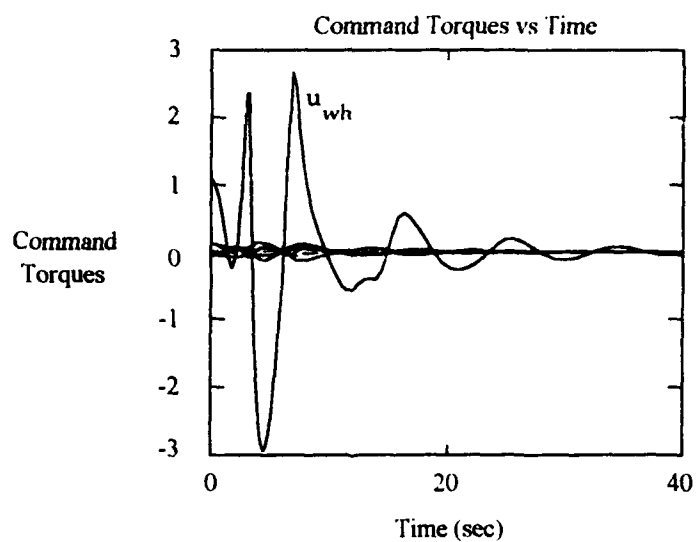
**Figure 27: Lyapunov Point Controller Angles**



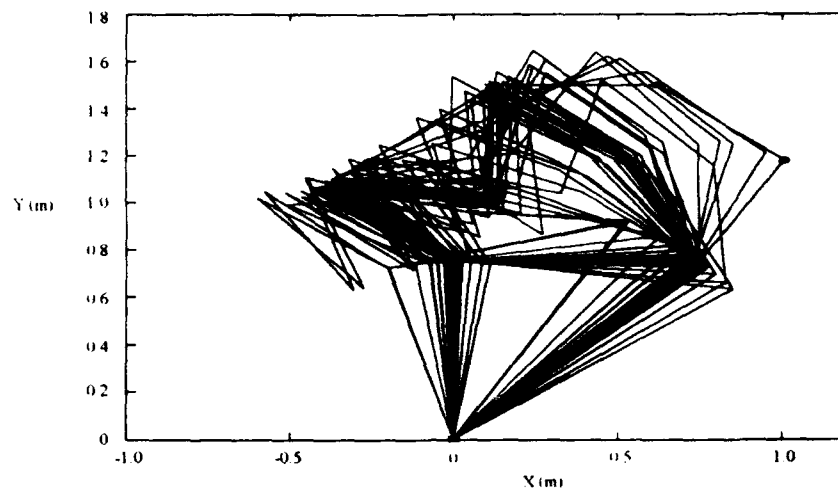
**Figure 28: Lyapunov Point Controller Displacement Errors**



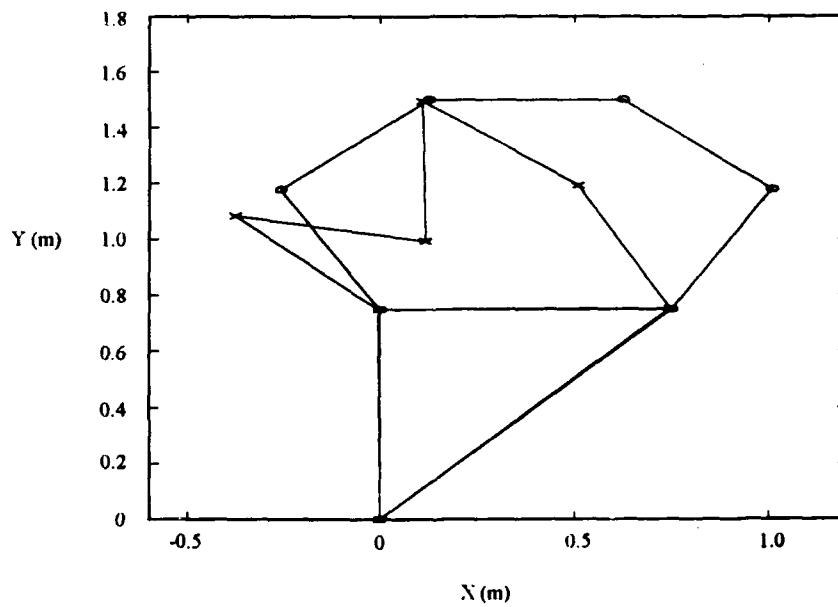
**Figure 29: Lyapunov Point Controller Angular Rates**



**Figure 30: Lyapunov Point Controller Command Torques**



**Figure 31: Lyapunov Point Controller Time Lapse Stick Figure**



**Figure 32: Lyapunov Point Controller Initial and Final Stick Figures**

## 2. Lyapunov Tracking Controller

This controller uses the following equation to calculate control torques

$$u = C_1^{\dagger} \{ -K_v \delta \dot{q} + C_{1_{ref}} u_{ref} - (C_2 \dot{q} - C_{2_{ref}} \dot{q}_{ref}) - (C_3 - C_{3_{ref}}) - K_p \delta q \} \quad (209)$$

This equation was developed in the analytical chapter and repeated here for convenience. The command torques are based on errors with a reference trajectory. Reference torques which resulted from minimizing a weighted norm of the actuator torques associated with the reference trajectory are also included.

### a. 5<sup>th</sup> Order Reference Trajectory

In this simulation, a fifth order polynomial reference trajectory is applied to the payload generalized coordinates. The payload coordinates displacements, velocities, and accelerations resulting from this polynomial are depicted in Figure 33. When calculating the reference torques from the inverse kinematics, the six joint actuators are all weighted equally. The maneuver time is selected to last 10 seconds. As is demonstrated in Figures 34-36, the system successfully moves from initial conditions to desired final conditions. The displacement errors are less than  $10^{-4}$  deg (Figure 34). The command torques (Figure 37) are an order of magnitude smaller than for the previous simulation which lacked a reference trajectory. Evaluating  $\int |u_{wh}| dt$  leads to the dramatically improved value of 0.5746. More importantly, the centerbody attitude is maintained throughout the maneuver. Figure 38 shows the time lapse depiction of the maneuver.

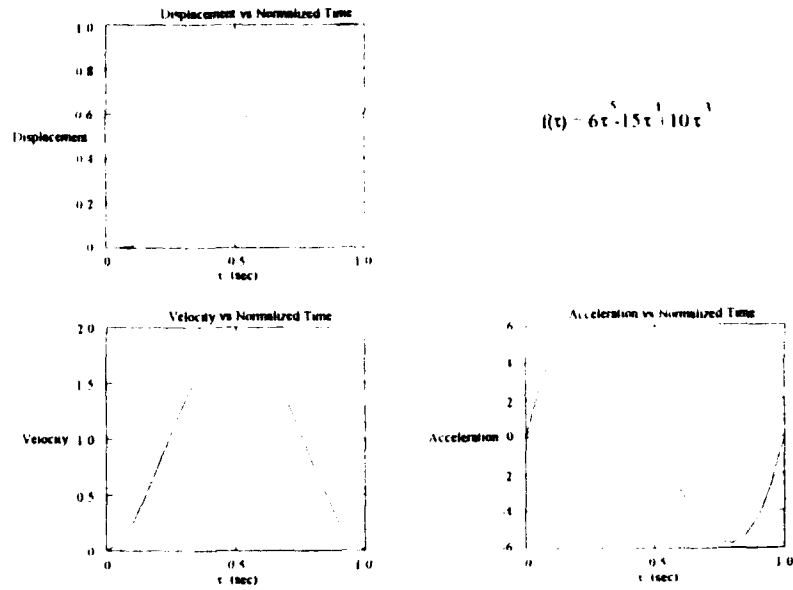


Figure 33: 5<sup>th</sup> Order Reference Trajectories

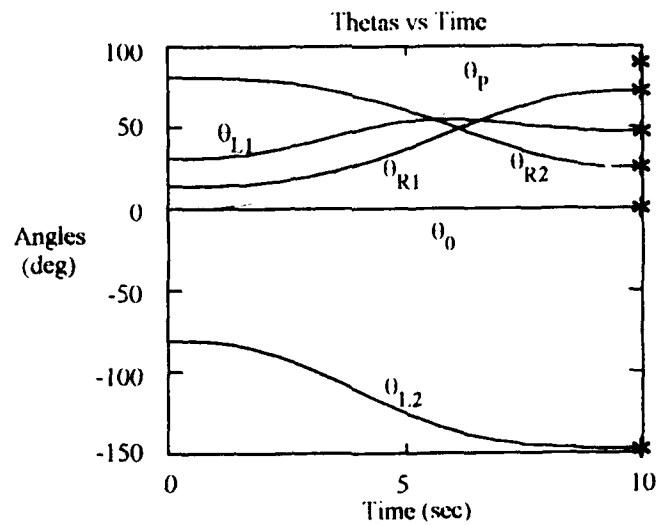


Figure 34: 5<sup>th</sup> Order Angles



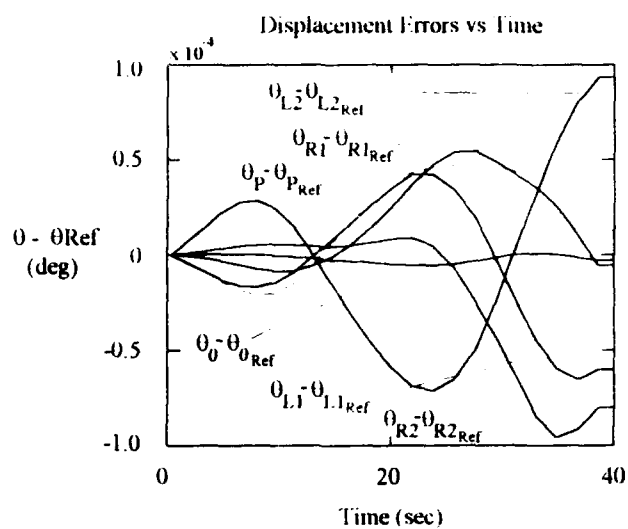


Figure 35: 5<sup>th</sup> Order Displacement Errors

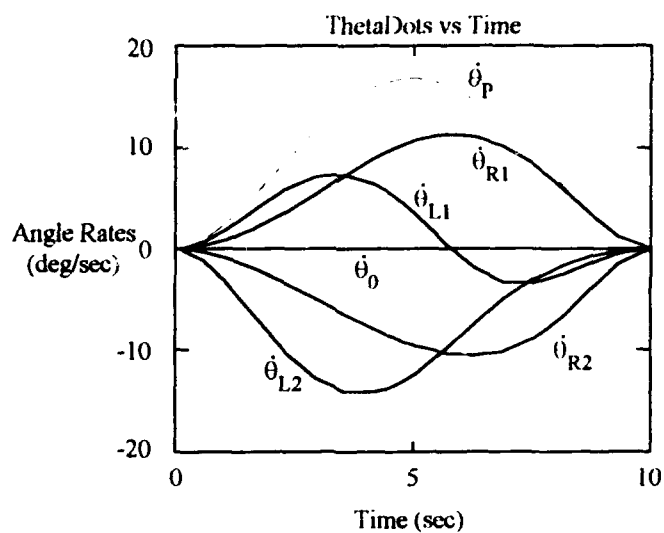
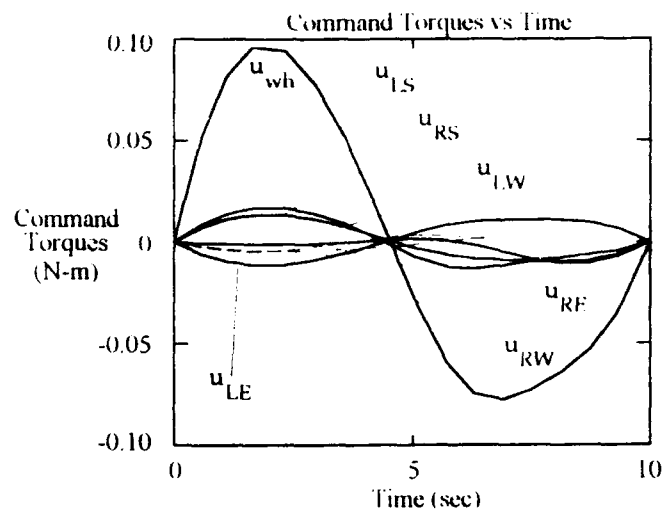
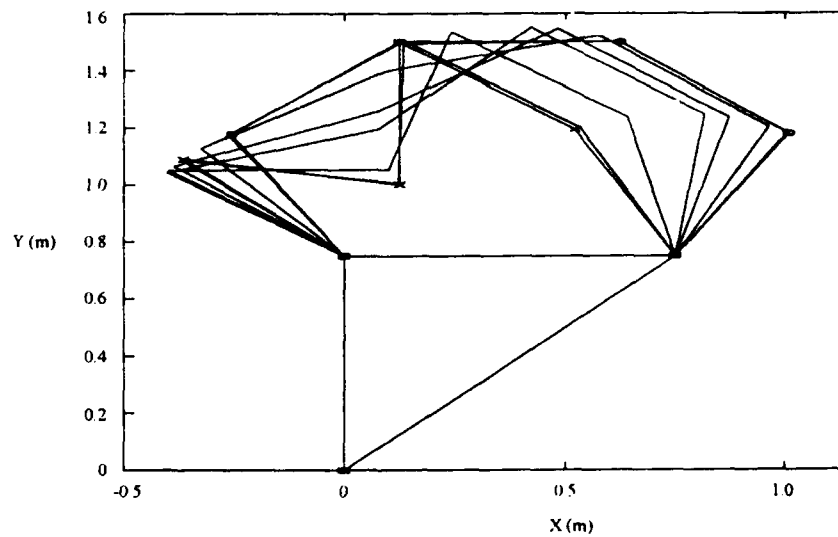


Figure 36: 5<sup>th</sup> Order Angular Rates



**Figure 37: 5<sup>th</sup> Order Command Torques**



**Figure 38: 5<sup>th</sup> Order Time Lapse Stick Figure**

*b. 8<sup>th</sup> Order Reference Trajectory*

By increasing the order of the reference trajectory polynomial while maintaining the same boundary conditions concerning velocity and acceleration, one hopes to achieve improved performance. For example, the domain of all sixth order polynomial functions includes all fifth order polynomial functions as a subset. Therefore, when searching all sixth order polynomials for coefficients which will minimize the cost function, one possible solution is the fifth order polynomial already used. Using the function minimization routine discussed in the previous chapter, a sixth order polynomial function was found. Although there was some improvement, the change in performance was not significant. The same was true for a seventh order function. An eighth order function is presented here. It was hoped that the increased order would be enough of a departure from the fifth order cause to produce significant improvement in reducing the centerbody disturbance torque. The algorithm converged to a solution for the eighth order polynomial after running approximately two hours on a personal computer with an Intel 486-DX50 cpu. The resulting trajectories are very similar to those for the fifth order case and are displayed in Figure 39. The most obvious difference is a lack of symmetry. Plots for this case are contained in Figures 40-43. The value of  $\int |u_{wh}| dt$  for this case was 0.5705.

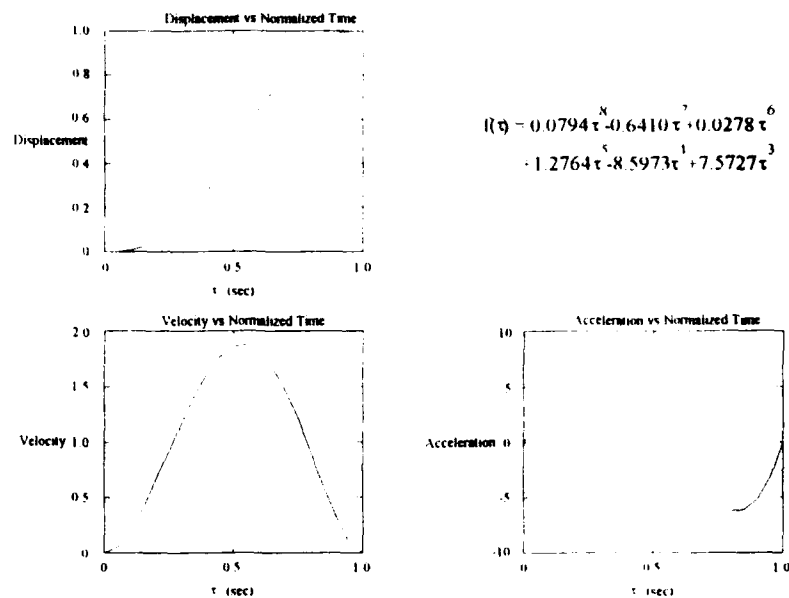


Figure 39: 8<sup>th</sup> Order Reference Trajectories

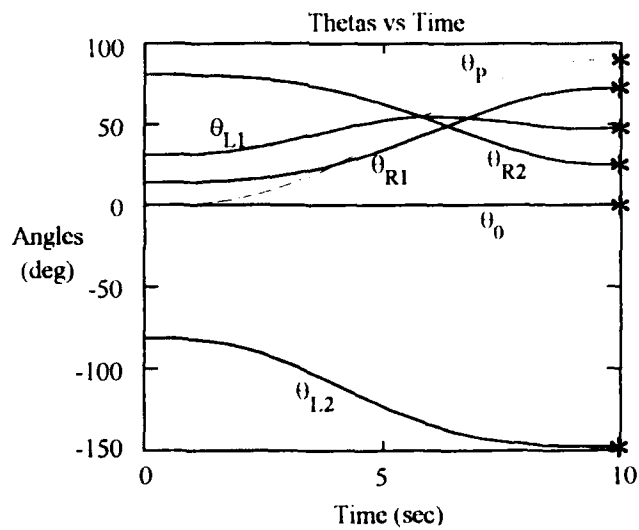
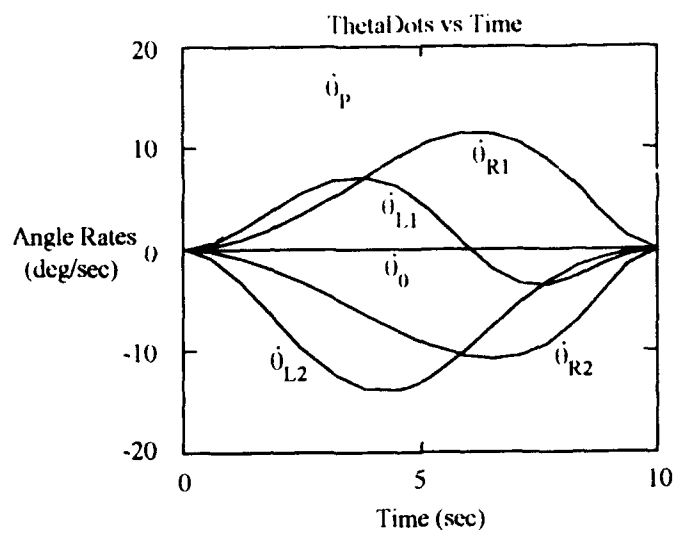
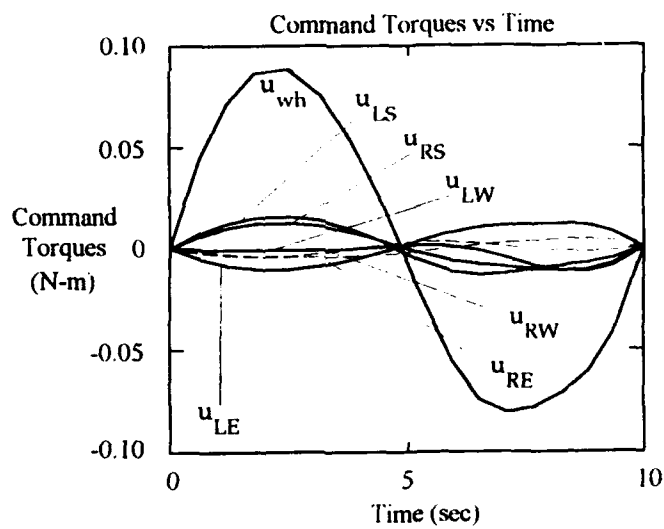


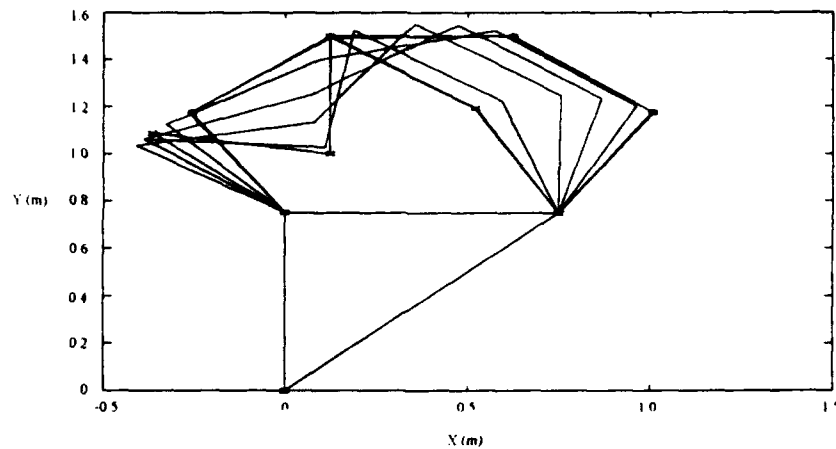
Figure 40: 8<sup>th</sup> Order Angles



**Figure 41: 8<sup>th</sup> Order Angular Rates**



**Figure 42: 8<sup>th</sup> Order Command Torques**

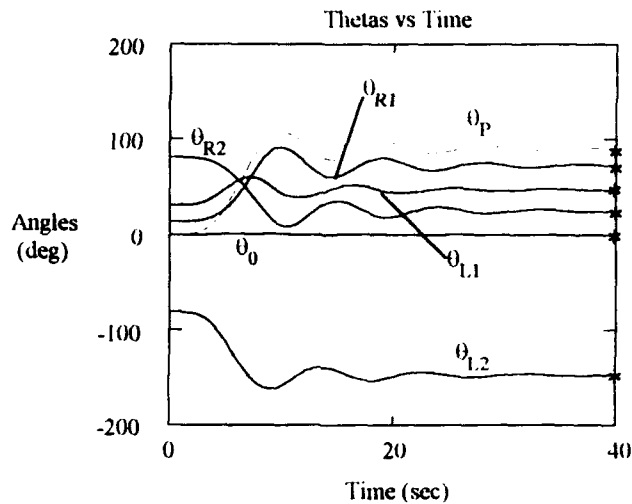


**Figure 43: 8<sup>th</sup> Order Time Lapse Stick Figure**

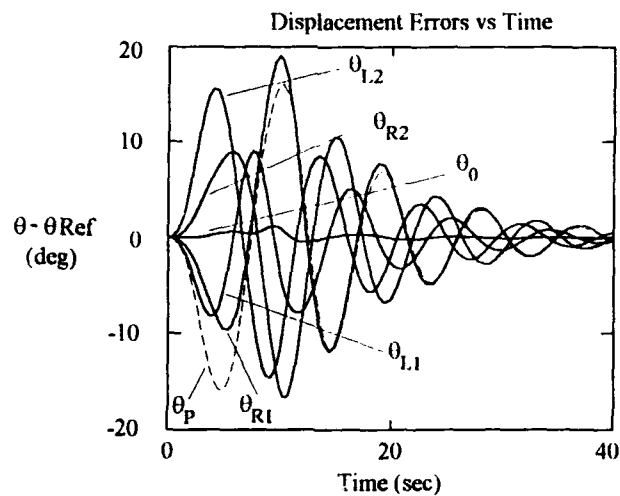
### **3. Modified Lyapunov Tracking Controller**

This simulation represents a compromise between the Lyapunov point controller and the Lyapunov tracking controller. Because the Lyapunov point controller does not use a reference trajectory, the cost function which minimizes the weighted norm of the actuator torques is completely bypassed. The modified Lyapunov controller removes the reference torque term from the command torque calculation (Eqn 209) but calculates command torques based on errors with a reference trajectory. Like the Lyapunov point controller, the modified Lyapunov tracking controller does not minimize a weighted norm of the actuator torques and is therefore not a cooperative controller. The angle histories in Figure 44 exhibit less of the oscillatory nature than the point controller simulation, but the accuracy shown in Figure 45 is considerably worse than the reference trajectory simulations. Figures 46-48 also illustrate behavior better than the point controller but not as good as when command torques are found using Eqn 209. The magnitude of the command torques show an order of magnitude improvement over the point controller. This is directly attributable to using intermediate reference points on the way to a desired final state rather than

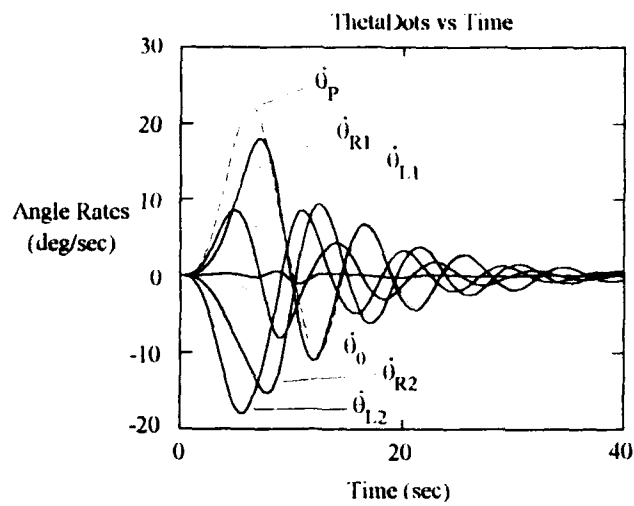
attempting to achieve the desired final state all at once. Calculating  $\int |u_{wh}| dt$  produced a value of 2.4523. The time lapse figure shows that the motion is much less wild but the centerbody attitude error is still noticeable.



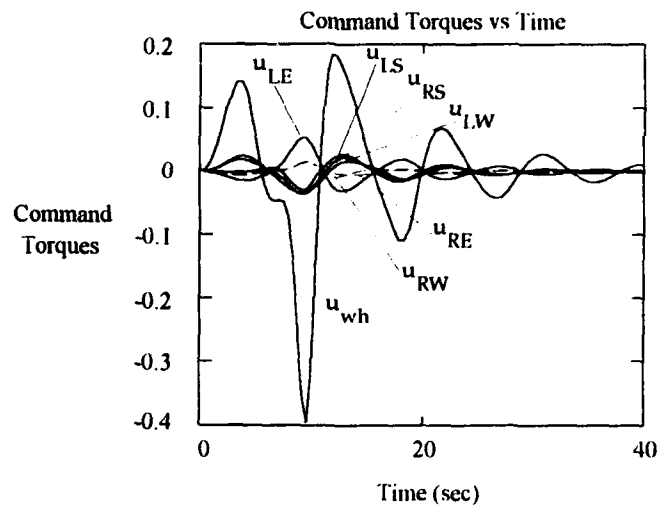
**Figure 44: Modified Lyapunov Tracking Controller Angles**



**Figure 45: Modified Lyapunov Tracking Controller Displacement Errors**

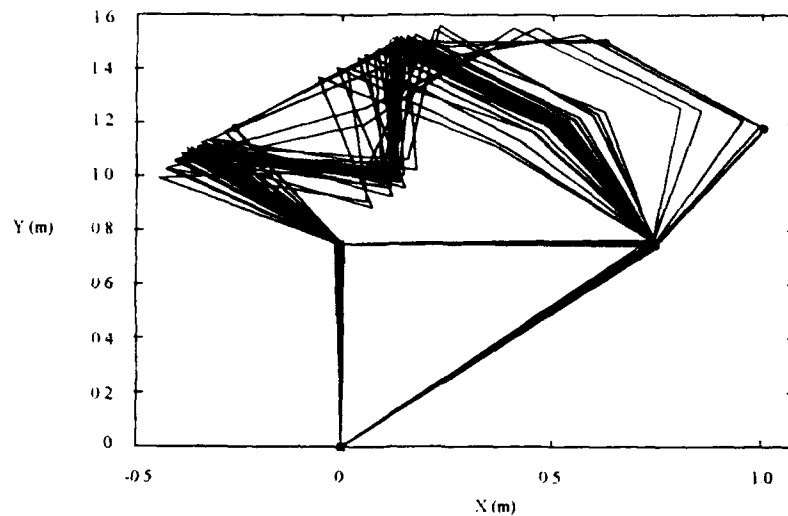


**Figure 46: Modified Lyapunov Tracking Controller Angular Rates**



**Figure 47: Modified Lyapunov Tracking Controller Command Torques**





**Figure 48: Modified Lyapunov Tracking Controller Time Lapse Stick Figure**

#### **4. Comparison of Controllers**

Table 2 summarizes the results of the Lyapunov point controller, the two Lyapunov tracking controller cases, and the modified Lyapunov tracking controller. The point controller clearly has the worst performance with high reaction wheel torque and large centerbody attitude error. The tracking controller performs much better. Reaction wheel torque is greatly reduced and centerbody attitude error is eliminated. As expected, increasing the order of the polynomial reduces the reaction wheel torque further, but the improvement is relatively small. The modified tracking controller strikes a compromise between the point controller and the tracking controller.

**TABLE 2. COMPARISON OF HYPOTHETICAL MODEL SIMULATIONS**

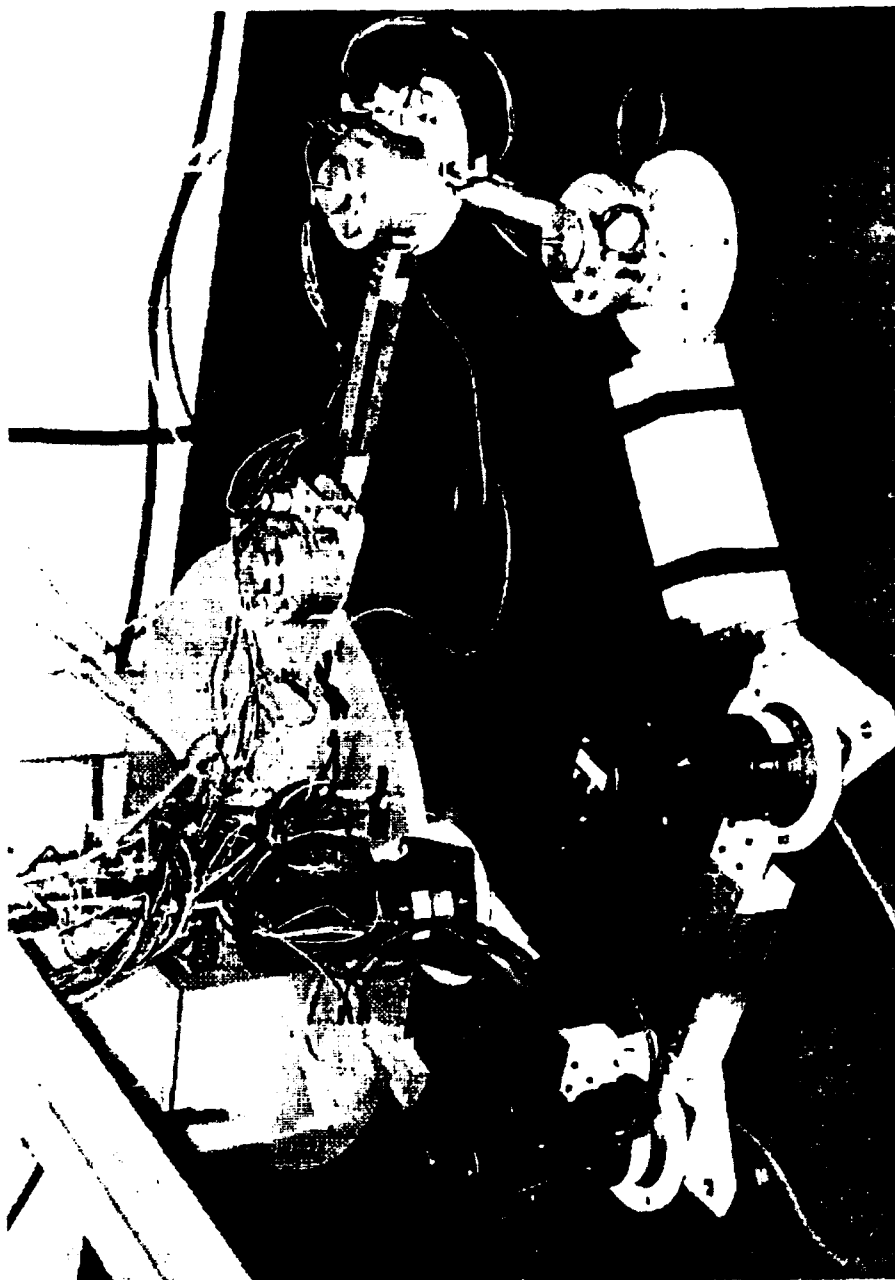
		$\int  u_{wh}  dt$	$ u_{max} $	Centerbody Attitude Error (deg)	Cooperative
Point Controller		17.3841	2.9365	16.2261	No
Tracking Controller	5 <sup>th</sup> Order	0.5746	0.0961	0.0000	Yes
	8 <sup>th</sup> Order	0.5705	0.0885	0.0000	Yes
Modified Tracking Controlier		2.4523	0.3950	1.1910	No

## **IV. EXPERIMENTAL WORK**

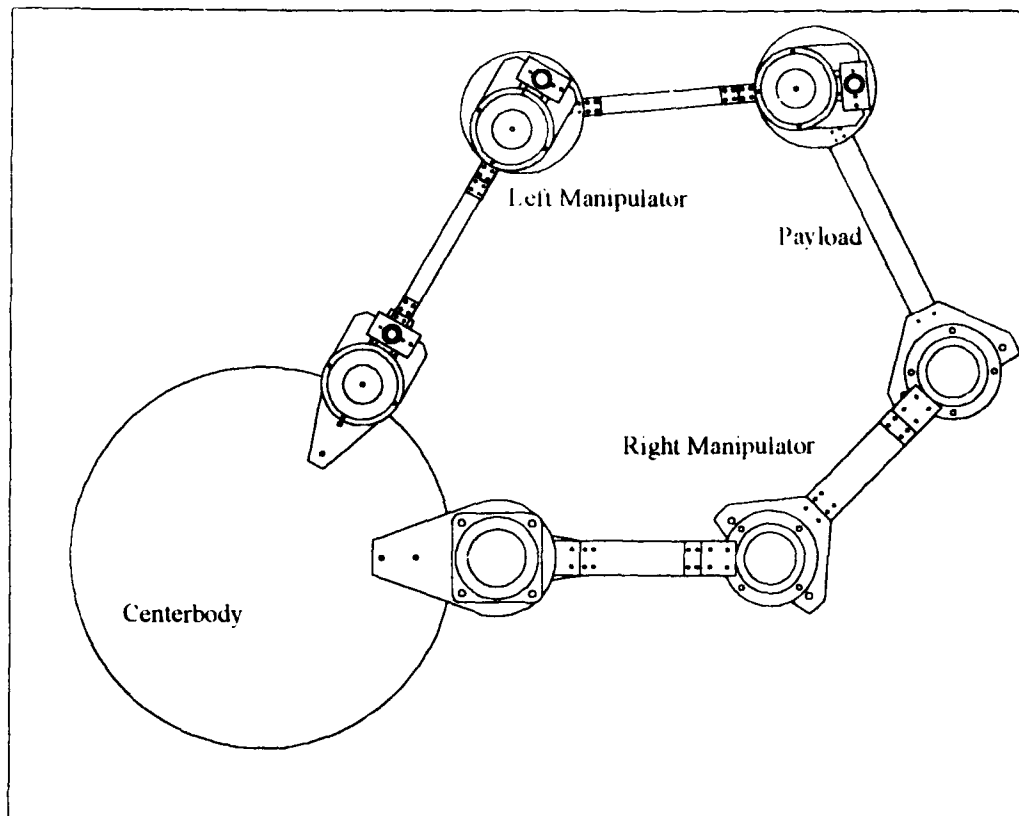
The experimental phase of this research was conducted on the Spacecraft Robotics Simulator (SRS). The SRS is a derivative of the Flexible Spacecraft Simulator (FSS) initially developed by Watkins [Ref 17] and later modified by Hailey [Ref 18]. Sorensen [Ref 18] began the work to convert the FSS into the SRS.

### **A. SETUP**

The SRS permits experimental investigation of two dimensional robotics motion and rotational spacecraft dynamics. The SRS is illustrated in Figures 49 and 50. The simulator hardware is floated on an eight foot by six foot granite table by means of a thin layer of air supplied by an external source. The table is polished to within 0.001 inch peak to valley and leveled to prevent gravitational accelerations from impacting the motion across its surface. The following sections describe the simulated spacecraft with its associated sensors and actuators and the controller which together form the SRS. The spacecraft components are the centerbody, two manipulators, and a payload.



**Figure 49: Spacecraft Robotics Simulator**



**Figure 50: System Top View**

### **1. Centerbody**

The centerbody is a 30 inch diameter, 7/8 inch thick aluminum disk. The centerbody carries a position sensor, rate sensor, momentum wheel, thrusters, and an air tank to power the thrusters. The centerbody also serves as the mounting platform for the manipulators. The centerbody is floated by a central air bearing and three air pads located at 120 degree intervals near the outer edge. The air pads are each capable of floating 60 pounds when the air pressure supplied to the pads is 80 psi. The air bearing is attached to an overhead I-beam which restricts to motion of the centerbody to rotation only.

Centerbody angular position is sensed by a Rotary Variable Displacement Transducer (RVDT) mounted directly above the air bearing. The RVDT is a model R30D manufactured by Schaevitz Sensing Systems. Its linear range is restricted to  $\pm 40$  degrees. Centerbody angular rate is measured by a rate transducer manufactured by Humphrey, Inc. The instrument has a range of  $\pm 100$  deg/sec and a minimum threshold of 0.01 deg/sec.

Centerbody angular position is controlled by a momentum wheel. The momentum wheel speed is measured by a tachometer contained in the servo motor which drives the momentum wheel. The centerbody momentum wheel is powered by a model JR16M4CH/F9T servo motor manufactured by PMI Industries. Characteristics of this motor are summarized in Table 3. Although the centerbody also carries two thrusters, they are not used in this research.

**TABLE 3. MOMENTUM WHEEL MOTOR CHARACTERISTICS**

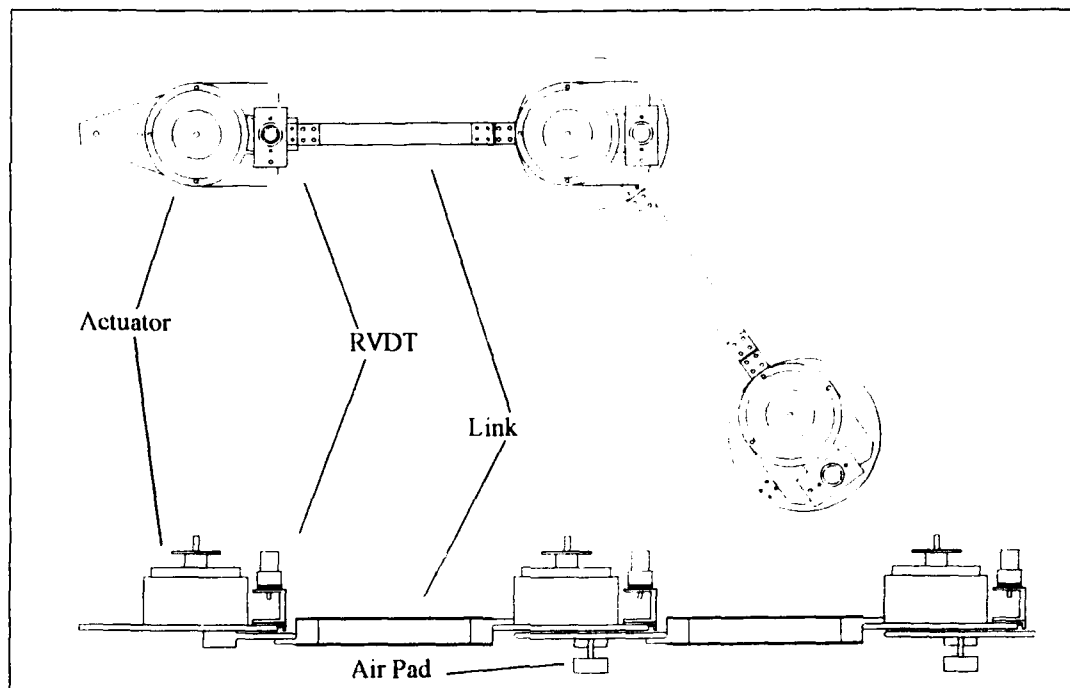
Manufacturer	PMI Industries
Model	JR16M4CH/F9T
Rated Output Speed (rpm)	3000
Rated Current (amps)	7.79
Rated Voltage (volts)	168
Rated Torque (in-lb)	31.85
Height (in)	4.5
Weight (lb)	17.5
Outside Diameter (in)	7.4

## **2. Manipulators**

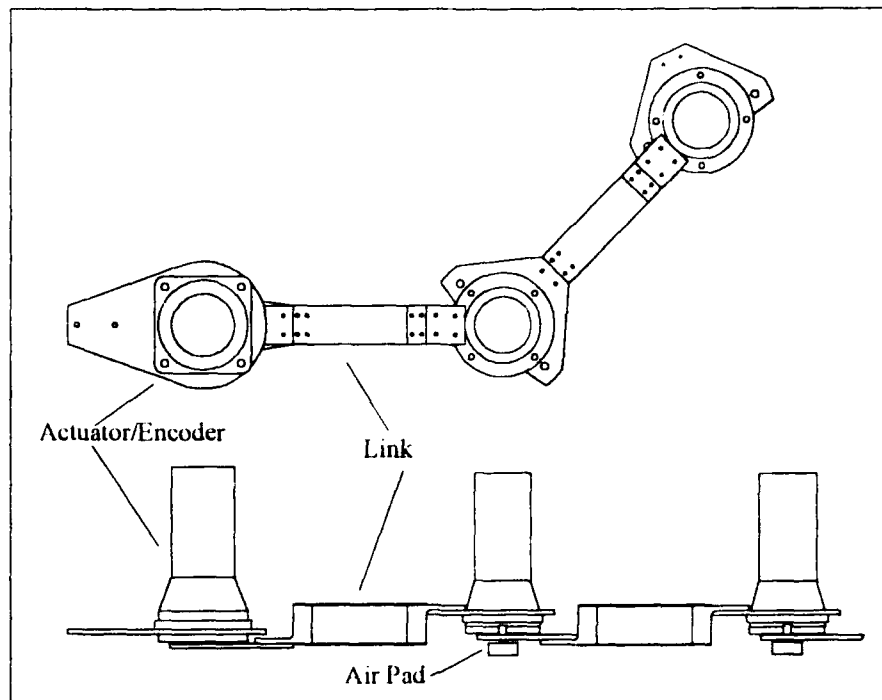
Two two-link manipulators are mounted 60 degrees apart on the centerbody. Each manipulator has three joints. The shoulder joints are supported by the centerbody while the elbow and wrist joints are supported by two air pads apiece. The links between the

joints are stiff laterally but permit some flexibility vertically. This feature increases the tolerances on the air pad height adjustment.

Left arm joint angles are measured by the same model RVDT as is used on the centerbody. All three of the left arm actuators are series 9FGHD servo disk motors manufactured by PMI Industries. Joint angles on the right arm are sensed by encoders purchased with the joint actuators. The encoder resolution is 0.005 degrees. The right arm joint actuators are harmonic drive dc servo actuators manufactured by HD Systems, Inc. The shoulder actuator is model RFS-25-6018-E036AL while the elbow and wrist actuators are model RFS-20-6012-E036AL. Specifications for the three types of joint actuators are contained in Table 4.



**Figure 51: Left Manipulator Top and Side Views**



**Figure 52: Right Manipulator Top and Side Views**

**TABLE 4. MANIPULATOR ACTUATOR CHARACTERISTICS**

Manufacturer	HD Systems	HD Systems	PMI Industries
Model	RFS-25-6012	RFS-25-6018	9FGHD
Reduction Ratio	1:50	1:50	1:148.5
Rated Output Speed (rpm)	60	60	17
Rated Current (amps)	2.9	3.9	5.6
Rated Voltage (volts)	75	75	12
Rated Torque (in-lb)	174	260	80
Height (in)	8.8	9.6	3
Weight (lb)	9.3	14.1	3.2
Footprint (in)	4.3 <sup>(1)</sup>	5.1 <sup>(1)</sup>	4.8 <sup>(2)</sup>

<sup>1</sup> Side of square

<sup>2</sup> Diameter of circle



The joint actuators are all driven by Kepco power supplies. These bipolar, programmable, linear amplifiers can be controlled manually from the front panel or controlled remotely with a  $\pm 10$  volt signal. In this application, the power supplies are operated in the current control mode with the voltage and current limits manually set consistent with the values in Table 4. The specific power supply models and their characteristics are summarized in Table 5.

**TABLE 5. POWER SUPPLIES CHARACTERISTICS**

Model	BOP 72-6M	BOP 72-3M	BOP 20-10M
Actuators Controlled	Right Shoulder	Right Elbow, Right Wrist	All Left Arm Joints
DC Output Range	$\pm 72$ volts $\pm 6$ amps	$\pm 72$ volts $\pm 3$ amps	$\pm 20$ volts $\pm 10$ amps
Closed Loop Gain	0.6 (amp/volt)	0.3 (amp/volt)	1.0 (amp/volt)

### **3. Payload**

The payload is a rigid bar mechanically fastened to the ends of both manipulators. The payload is supported entirely by the air pads on the manipulator wrist joints. Ballast can be added to the payload to change the mass and inertia characteristics of the system. This allows for the construction of cases in which the mass of the payload is nontrivial compared to the spacecraft centerbody. The payload contains no sensors or actuators. Payload position is derived from the manipulator joint angles.

### **4. Controller**

The AC-100 programmable controller manufactured by Integrated Systems, Inc. controls the SRS. The AC-100 includes an Intel 80386 Application Processor, an Intel 80386 Multibus II Input/Output Processor, an Intel 80386 Communication Processor, and

Intel 80387 Coprocessor, a Weitek 3167 Coprocessor, and Analog-To-Digital and Digital-To-Analog Data Translation DT2402 Input/Output Board, two INX-04 Encoder and Digital-To-Analog Servo Boards, and an Ethernet Interface Module. The AC-100 also includes software installed on a VAX 3100 Series Model 30 workstation. The software permits design of a controller in block diagram graphical form and conversion of the diagram to C language programming code. The user is also able to design an interactive animation window to operate the controller. The AC-100 receives input signals from the sensors and the graphical user interface. AC-100 output signals go to the power supplies driving the actuators or to the graphical user interface for display.

### **5. System Integration**

The differences between the ideal world of an analytical simulation and the real world of actual hardware became apparent during system integration. A few problems arose then requiring some modification of the experiment. The first problem concerned floating the centerbody. It exhibited a noticeable resistance to rotation. This is due in part to the air pressure of the available air supply. Because it was only 40 psi, the air pads performance was degraded by a factor of two. Prior to mounting the manipulators, the centerbody weighed approximately 125 lbs. Adding the shoulder motors increased the centerbody weight to 145 lbs. The extra weight may have been enough to overwhelm the centerbody air pads. A second contributing factor to the centerbody drag is the inability of the central air bearing to function except under very low lateral loading. The modification to the experiment created by the centerbody problem is to not float the centerbody.

A second problem involved using the RVDTs. As envisioned, the experiment requires one RVDT for the centerbody and three for the left manipulator joints. The Space Dynamics laboratory has a total of three in stock. Although a fourth has been ordered, it did not arrive in time to be used. Using the existing RVDTs revealed another problem. Data acquisition of the RVDT signal by the AC-100 exhibited a random toggling of the

sensed value between a good reading and a value of zero. Because the angle information is critical to calculating actuator commands, this behavior is unacceptable. Consultation with the Integrated Systems technical support group revealed that this type of behavior is a bug within the AC-100 software which has been corrected in more recent versions. Use of the newer version was not possible because it requires upgrading the VAX workstation hardware and an updated version of the VMS operating system. The experimental modification used to overcome these difficulties is to derive the joint angles and velocities of the left manipulator by using the sensed information from the right manipulator encoders. Velocities were not sensed directly but approximated by the change in displacement which occurred since the last sample divided by the sample rate

A third obstacle involved the limitations of software to design the control algorithm. The block diagram construction method did not permit convenient matrix operations. Matrix multiplication must be programmed in an element by element basis. Matrix inversion must also be calculated by constructing a series of blocks to find each element. This handicap is not serious when the system equations of motion are of low order. However, the dual two link manipulator configuration is eighth order and beyond the practical means of programming complex matrix operations, especially matrix inverses. Recall that the command torques are calculated by the following relationship

$$\underline{u} = (C_1^T C_1)^{-1} C_1^T \{ -K_v \delta \dot{q} + C_{1_{ref}} \underline{u}_{ref} - (C_2 \dot{q} - C_{2_{ref}} \dot{q}_{ref}) - (C_3 - C_{3_{ref}}) - K_p \delta q \} \quad (210)$$

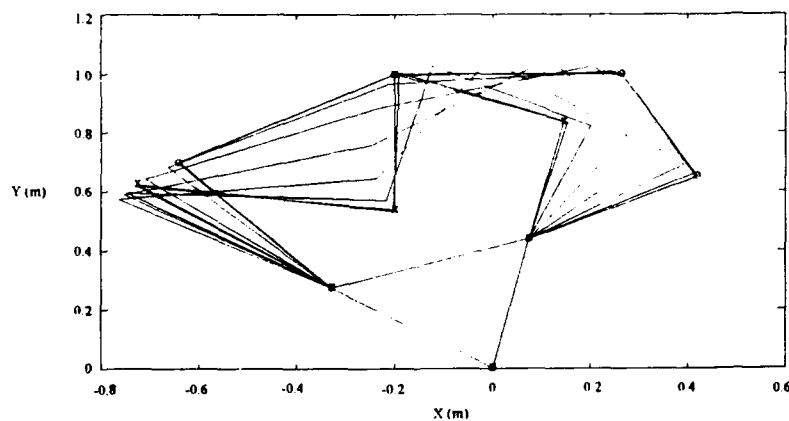
When the differences between the actual path and the reference path are small, this control law simplifies to something very similar to a PD controller. Therefore, the control law used by the experiment is a PD controller rather than the complete Lyapunov controller.

Performance differences between the left and right manipulator actuators also presented some problems. Because of the actuator redundancy, any three joint actuators should be enough to follow a reference trajectory. This fact can be demonstrated by using

only the three right joint actuators. However, the same trajectory is not possible with only the left actuators. The torque provided by the left joint actuators is insufficient to completely overcome the internal friction of the right joint actuators. Even when the left joint actuators are commanded manually from the front panel, there is no correction to reduce the position error. When steadily increasing the commanded current to the motor, the current limit is reached before the motor responds.

## B. RESULTS

The reference trajectory for the experimental phase is slightly different from that used in the analytical section. The reference maneuver still involves a 90 degree rotation of the payload with the right endpoint ending where the left endpoint began. The differences arise from the system parameter such as lengths and masses not being the same as in the generic hypothetical model. The desired reference maneuver is depicted in Figure 53. Results are shown in Figures 54-58 and summarized in Table 6. The sudden changes from believable values to zero in the figures are problems with the data acquisition software and do not indicate actual changes in the experimental hardware geometry.



**Figure 53: Desired Experimental Repositioning Maneuver**

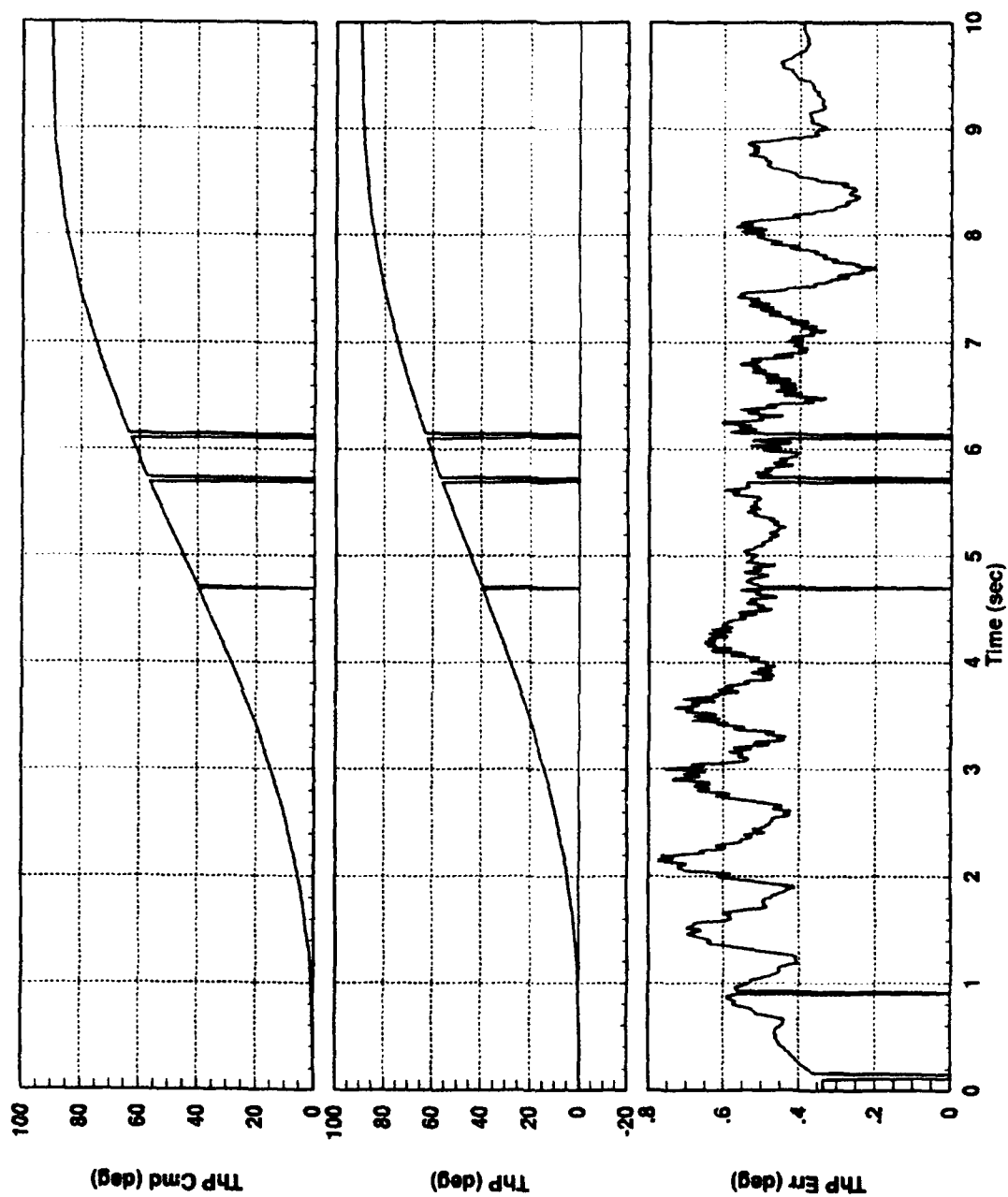


Figure 54:  $\theta_p$  Commanded, Actual, and Error Angles vs Time

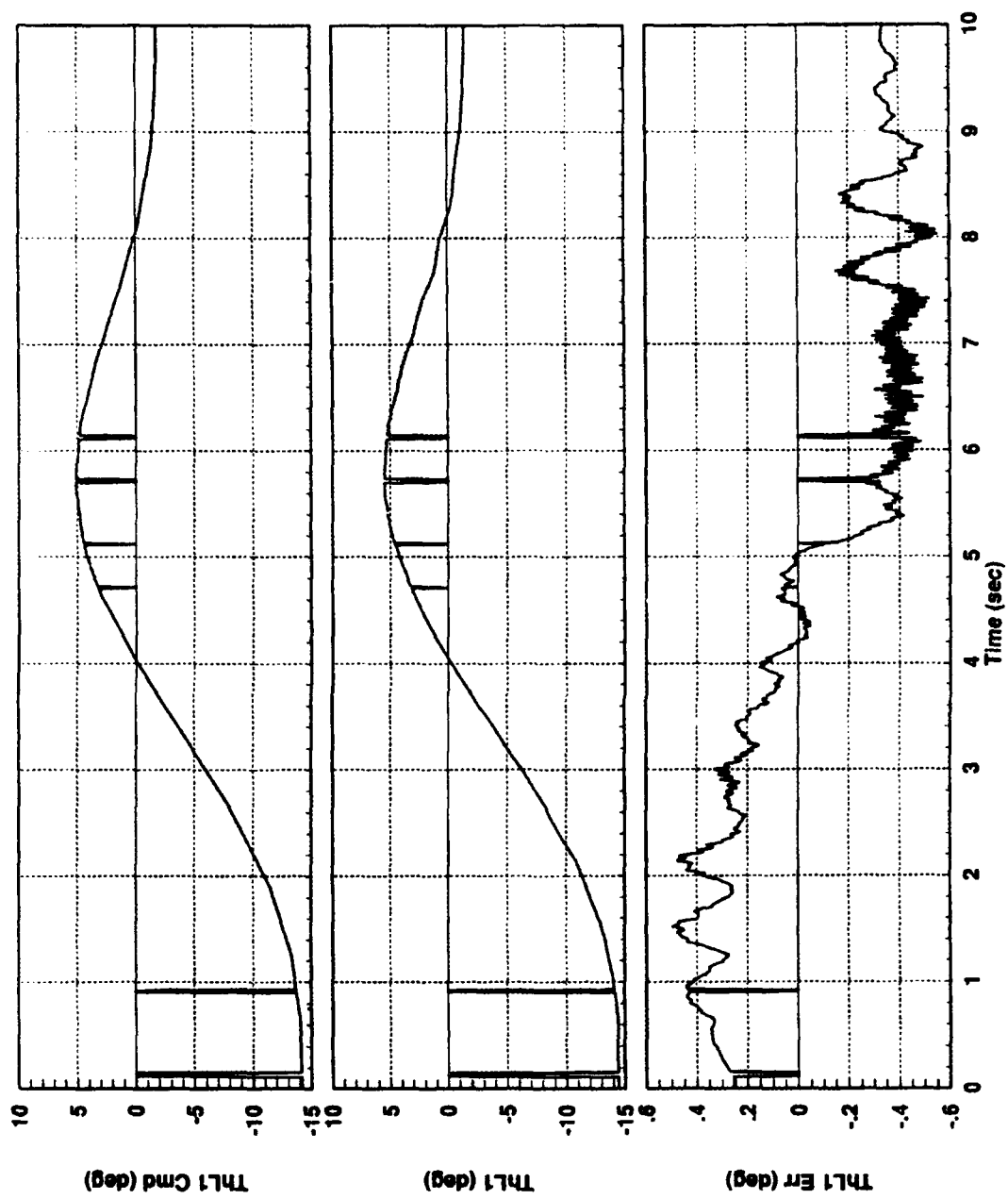


Figure 55:  $\theta_{L1}$  Commanded, Actual, and Error Angles vs Time

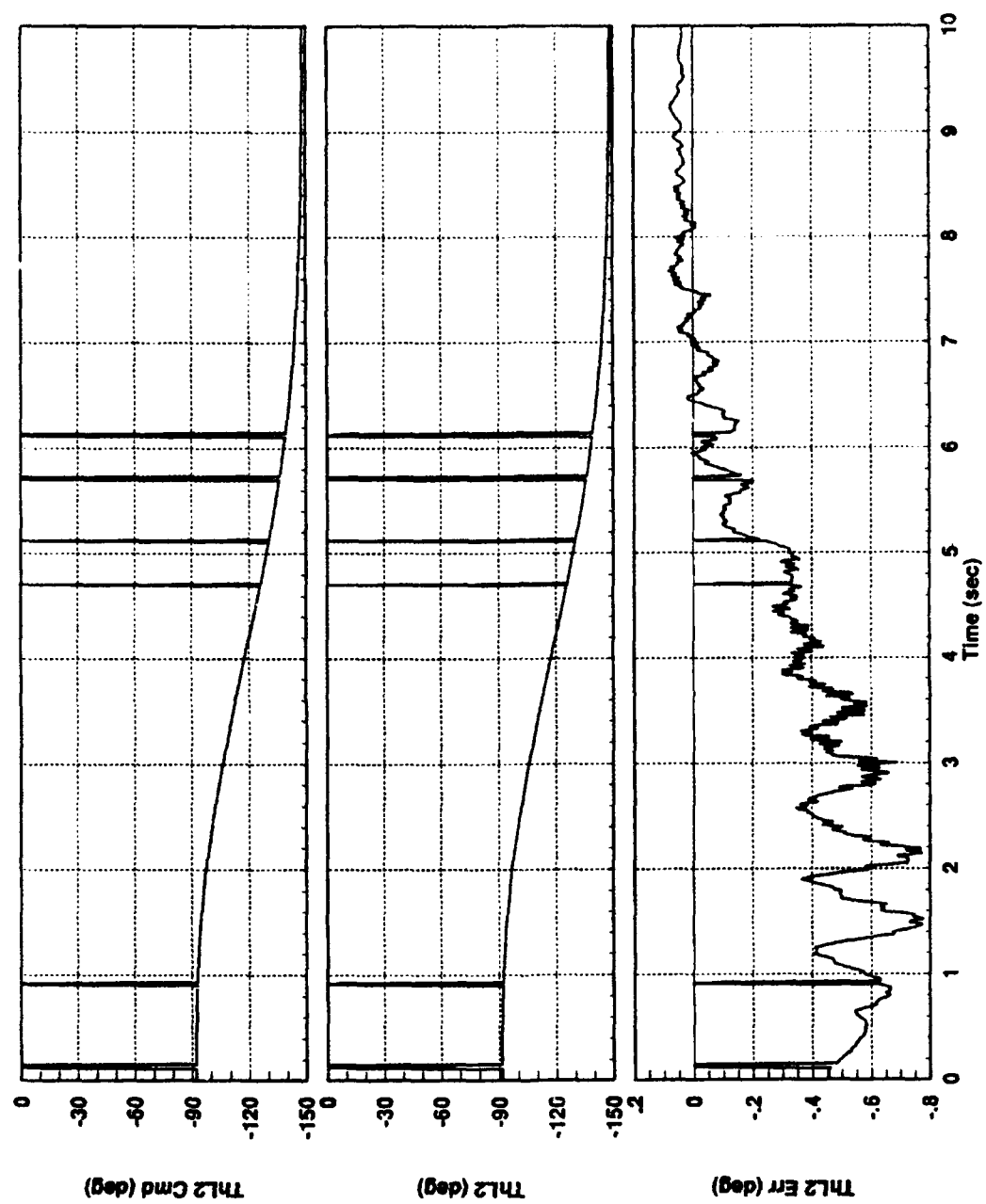


Figure 56:  $\theta_{L2}$  Commanded, Actual, and Error Angles vs Time

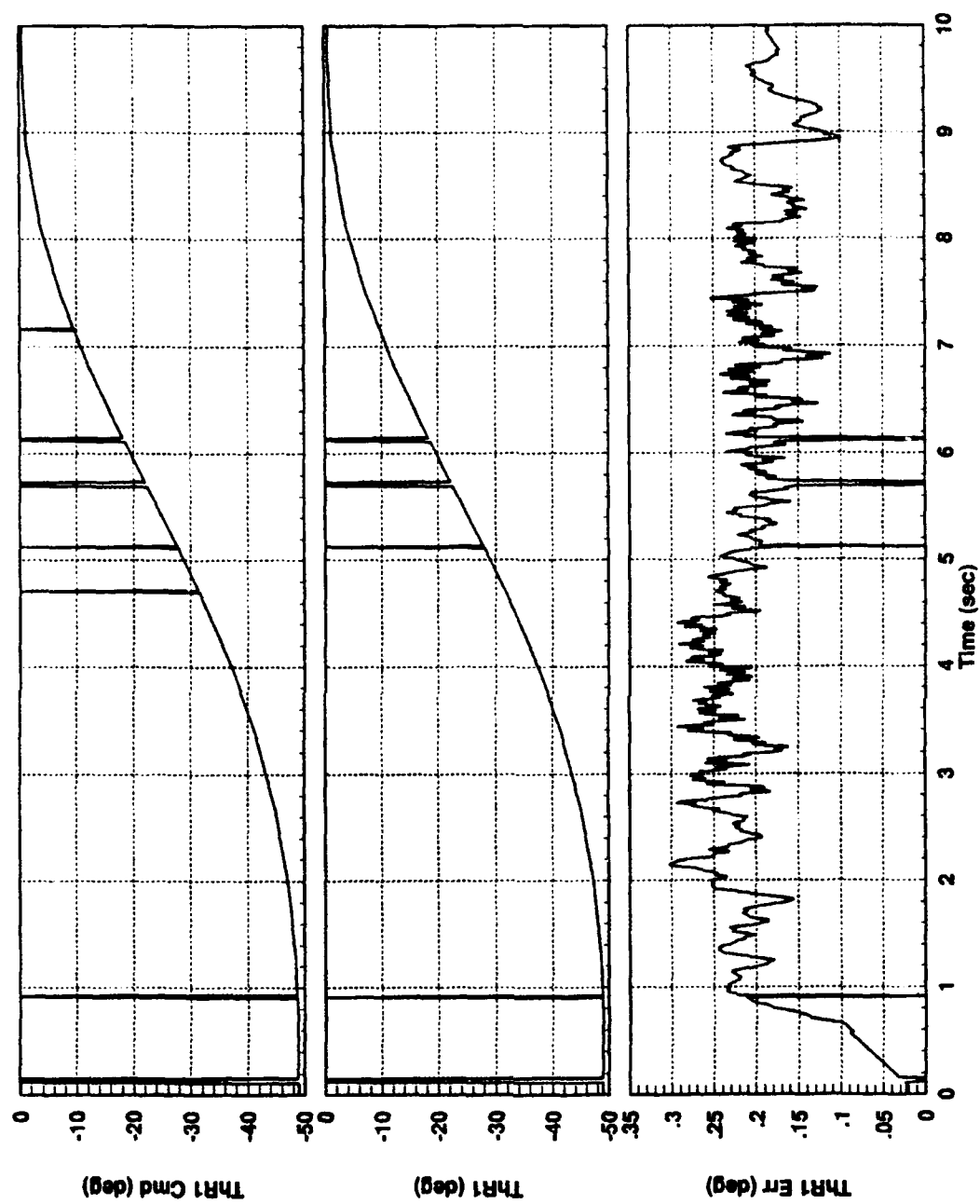


Figure 57:  $\theta_{R1}$  Commanded, Actual, and Error Angles vs Time



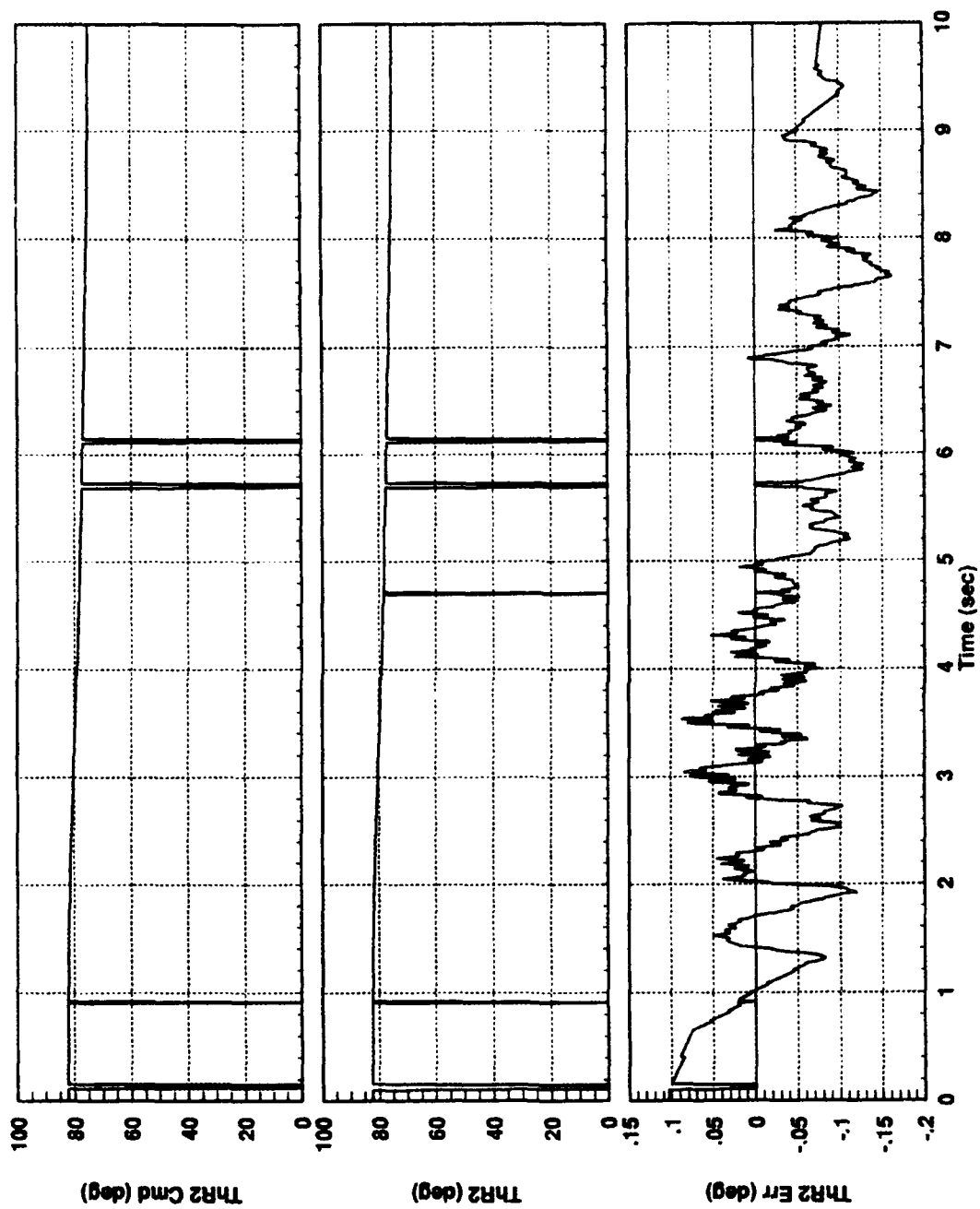


Figure 58:  $\theta_{R2}$  Commanded, Actual, and Error Angles vs Time

**TABLE 6. EXPERIMENTAL ERROR ANGLES**

	Errors (deg)		
	Initial	Final	Maximum Magnitude
$\theta_P$	0.2550	-0.3383	0.5527
$\theta_{L1}$	-0.4574	0.0366	0.7797
$\theta_{L2}$	0.0225	0.1873	0.3035
$\theta_{R1}$	0.1037	-0.0808	0.1628
$\theta_{R2}$	0.3350	0.3950	0.7742

## V. SUMMARY AND CONCLUSIONS

### A. SUMMARY

The dynamics of a dual two-link manipulator system which is repositioning an already grasped payload have been analyzed. The equations of motion for the system were developed using Lagrange's method. The resulting equations were highly nonlinear, coupled, second order differential equations. Given any reference trajectory, the actuator torques that will produce that trajectory were calculated to minimize a weighted norm of the torques. Stability of the system during the repositioning maneuver was ensured by a controller derived from Lyapunov stability theory. Equations for deriving joint angles from centerbody and payload reference values was also developed. Polynomial reference trajectories were presented as an attractive means to specify a reference trajectory.

The analytical model was validated using test cases in which some results could be predicted in advance. The model demonstrated conservation of energy when no torques were applied. It also exhibited conservation of angular momentum whenever the reaction wheel was disabled. The model also maintained symmetric geometry in the appropriate test cases. In cases which used the reaction wheel, conservation of energy and angular momentum did not apply. However, comparison of the change in angular momentum with the reaction wheel torque provided validation. Finally, in all test cases as well as simulations, the constraints were satisfied as measured by  $\Lambda \dot{q} + \Lambda_0 = 0$ .

Results from simulations indicated that the Lyapunov point controller, although stable, behaved poorly. Large centerbody attitude errors, high command torques, and wild oscillations make this controller undesirable for large repositioning maneuvers. The Lyapunov

tracking controller exhibited dramatic a improvement in performance. Centerbody attitude errors were removed and reaction wheel torque decreased significantly.

The experimental phase revealed that the controller required further simplification for compatibility with the laboratory resources. Acceptable results were obtained using a PD control law with a reference trajectory.

The objectives of this research were to 1) develop a stable control law that facilitates cooperation among the manipulators as they reposition the payload, 2) minimize the joint actuator effort, 3) reduce the disturbance torque transmitted to the spacecraft main body by the manipulator motion, and 4) validate the analytical development with experimental results. The Lyapunov controller satisfies the first objective. The second objective is achieved by the weighted norm calculation of the actuator torques. Reduction of the centerbody disturbance torque is accomplished through reference trajectory selection. Although a rigorous application of classical optimal control techniques proved impractical, a polynomial reference trajectory in which the coefficients were selected to reduce the disturbance torque was easily applied. Difficulties were encountered with regards to the fourth objective, experimental work. The controller developed analytically could not be directly transferred to the laboratory. This was due to a combination of hardware limitations and real world conditions instead of the ideal environment of the analytical model. The controller was adapted to the realities of the laboratory and resulted in successful accomplishment of a payload repositioning maneuver.

## **B. ORIGINAL CONTRIBUTIONS**

A simulation tool has been developed to analyze the dynamics of a space based robotics system. Some of the features of this tool include:

- (i) rotational motion of the spacecraft centerbody and planar motion of the manipulators and payload;
- (ii) minimization of a weighted norm of the actuator torques based on a user supplied weighting matrix;
- (iii) calculation of polynomial reference trajectory coefficients to produce a local minimum for the integral of the absolute value of the disturbance torque based on a user supplied order for the reference polynomial and an initial guess for the coefficients;
- (iv) a reference trajectory with zero velocity and acceleration at the beginning and end of the maneuver;
- (v) a Lyapunov controller which guarantees stability in the face of perturbations between the reference trajectory and the actual dynamics caused by errors in the initial conditions.

An experimental test bed was also developed. This effort involved the design of the manipulator components and the development of a real time controller. This test bed remains in the Spacecraft Dynamics and Control Laboratory and is available for follow-on work.

### **C. RECOMMENDATIONS FOR FURTHER STUDY**

As with any research, this work answers some questions but raises others. One of the areas that could receive further attention is the selection of the Lyapunov controller gains. The theory requires positive definite matrices to ensure stability but offers no insights concerning selection of the matrices to improve performance. For any given set of controller matrices, one expects the relative merits of the point controller, tracking controller, and

modified tracking controller to remain the same. However, still better performance might be achieved across the board if the gains were optimized.

Rather than merely changing Lyapunov controller gains, one might investigate another Lyapunov controller by beginning with a different candidate Lyapunov function than the one presented here. The choices are infinite and the results and performance difficult to predict.

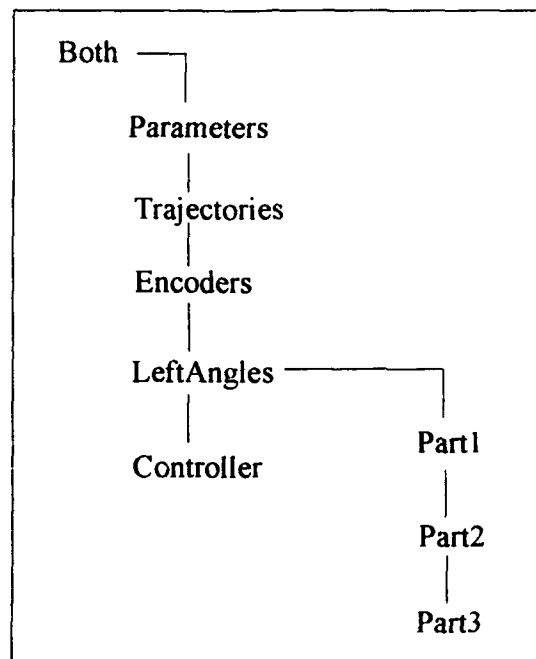
Trajectory optimization is another area that would benefit from further work. The function minimization algorithm used to select polynomial coefficients converged to local minima solutions depending on the initial guess for the coefficients. The search for a global minimum for a particular order polynomial requires further investigation. An alternate approach with respect to trajectory optimization is to use some function other than a simple polynomial to describe the trajectory. Possible trajectories might be Tchebycheff polynomials, Legendre polynomials, or Fourier series.

To help bridge the gap between the analytical model and the real world hardware, one could consider modifying the controller to include joint friction, actuator backlash, sensor noise, and flexibility. One could also consider using a minimum generalized coordinate formulation. One might also attack the differences from the hardware perspective by seeking components that more closely resemble those in the analytical model. Another improvement in the experiment would be to replace the existing joint velocity approximations with either an observer or an actual velocity measurement.

Finally, it's a three dimensional world. Extending the analytical model and, if possible, the laboratory experiment to include out of plane motion should be considered.

## APPENDIX A: EXPERIMENTAL CONTROL BLOCK DIAGRAMS

This appendix includes the block diagrams of the System Build super blocks made to control the SRS. The hierarchy among the super blocks is illustrated in Figure 59. Both is the parent superblock. The others are lower level super blocks.

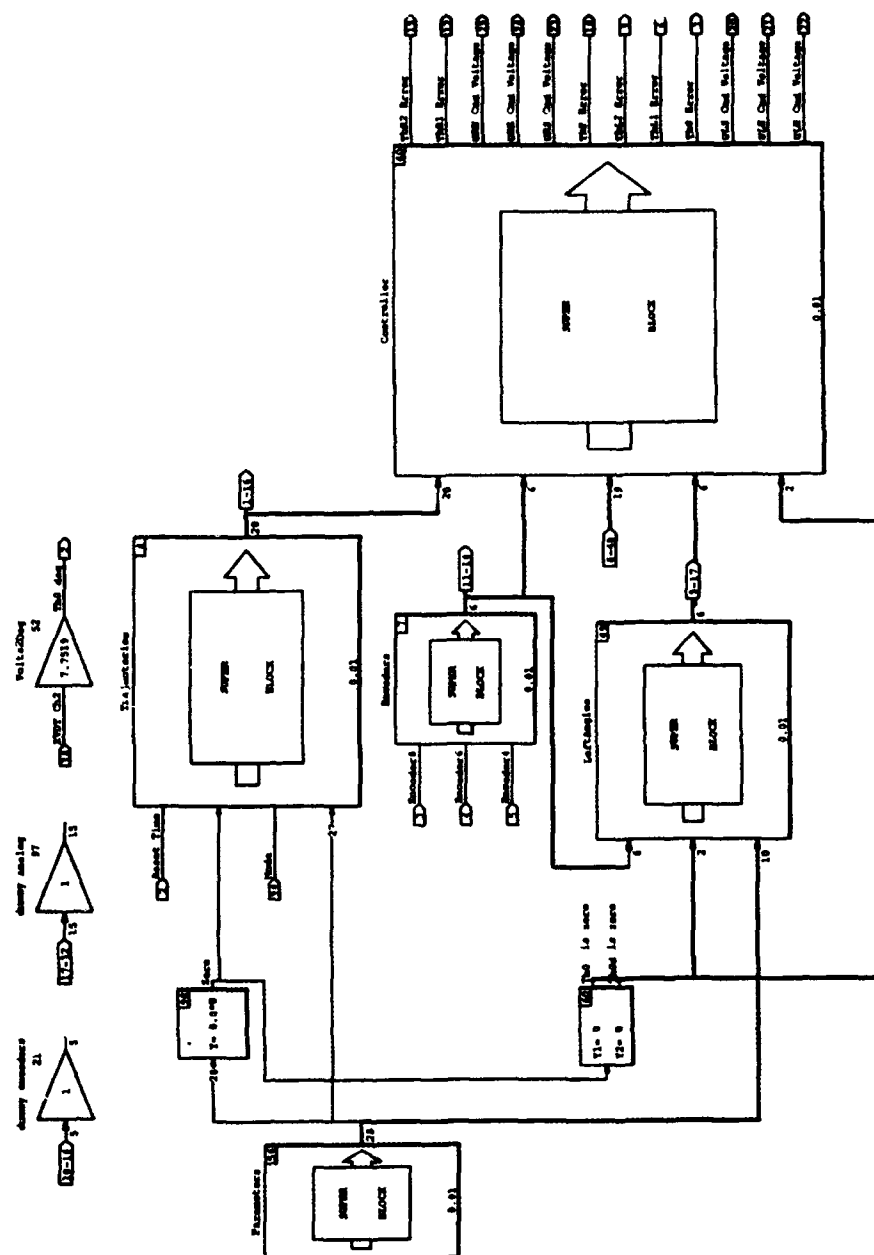


**Figure 59: Super Blocks Hierarchy**

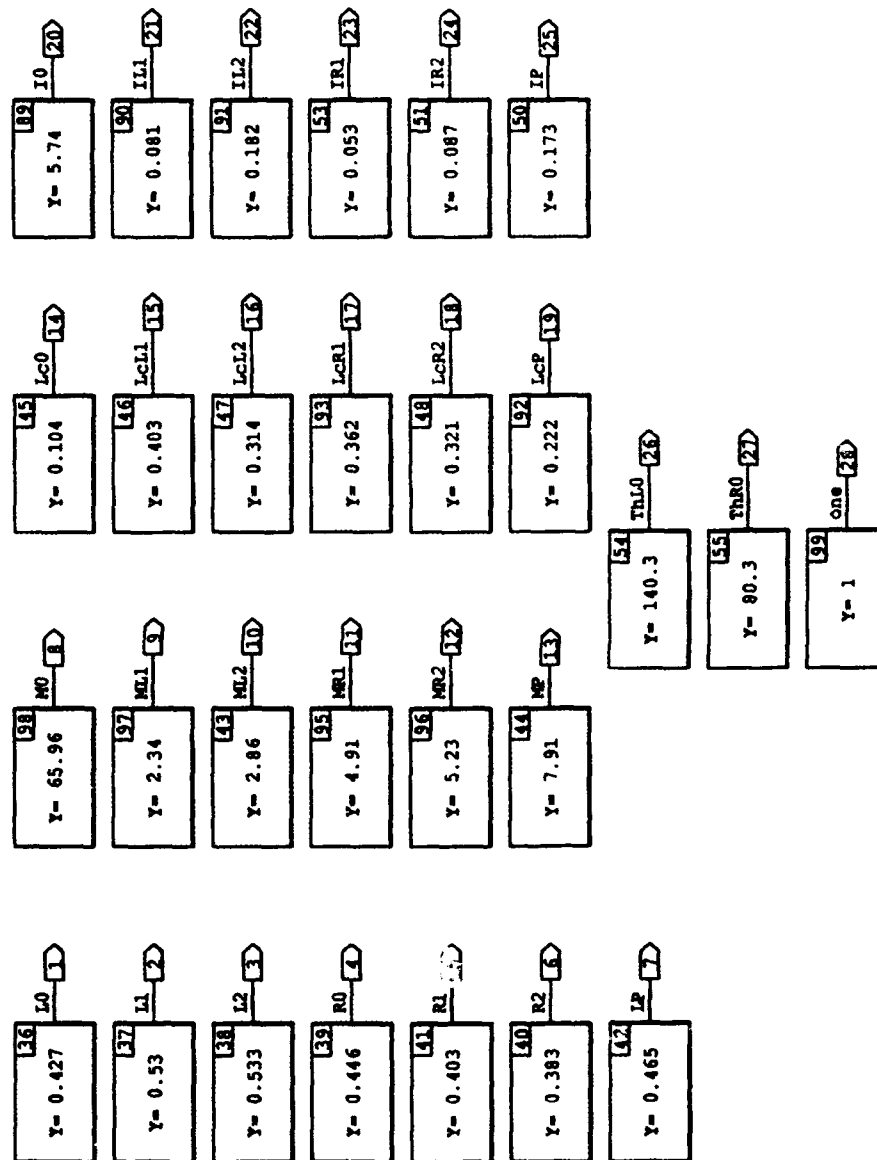
The block diagram for super block Both is shown in Figure 60. Inputs into the diagram include the sensor signals from the hardware and user operated dials to select the controller gains and enable switches which select the combination of joint actuators to enable. The outputs include commanded, reference, and error signals for each of the centerbody angle, joint angles, and payload angle. Motor current commands to the Kepco

power supplies are also outputs. Block 56 contains the system parameter values for the experimental hardware. This block is expanded and displayed in Figure 61. Block 8 contains the position and velocity values for a reference trajectory in a look-up table. It also contains a table to reset the system back to its original geometry to permit rerunning the reference trajectory. This block is expanded in Figure 62. Conversion of the encoder pulses from the right manipulator into angle and angle rate information is done in Block 7 which is expanded in Figure 64. Conversion of the encoder pulses from the right manipulator into angle information for the left manipulator is done in Block 49. Details of this block are shown in Figures 64-64. The PD controllers which convert the error signals into actuator commands are in Block 40. This block is expanded in two parts. The actuator commands for the right and left manipulator are shown in Figures 68 and 69 respectively.





99



100

27-JUL-93

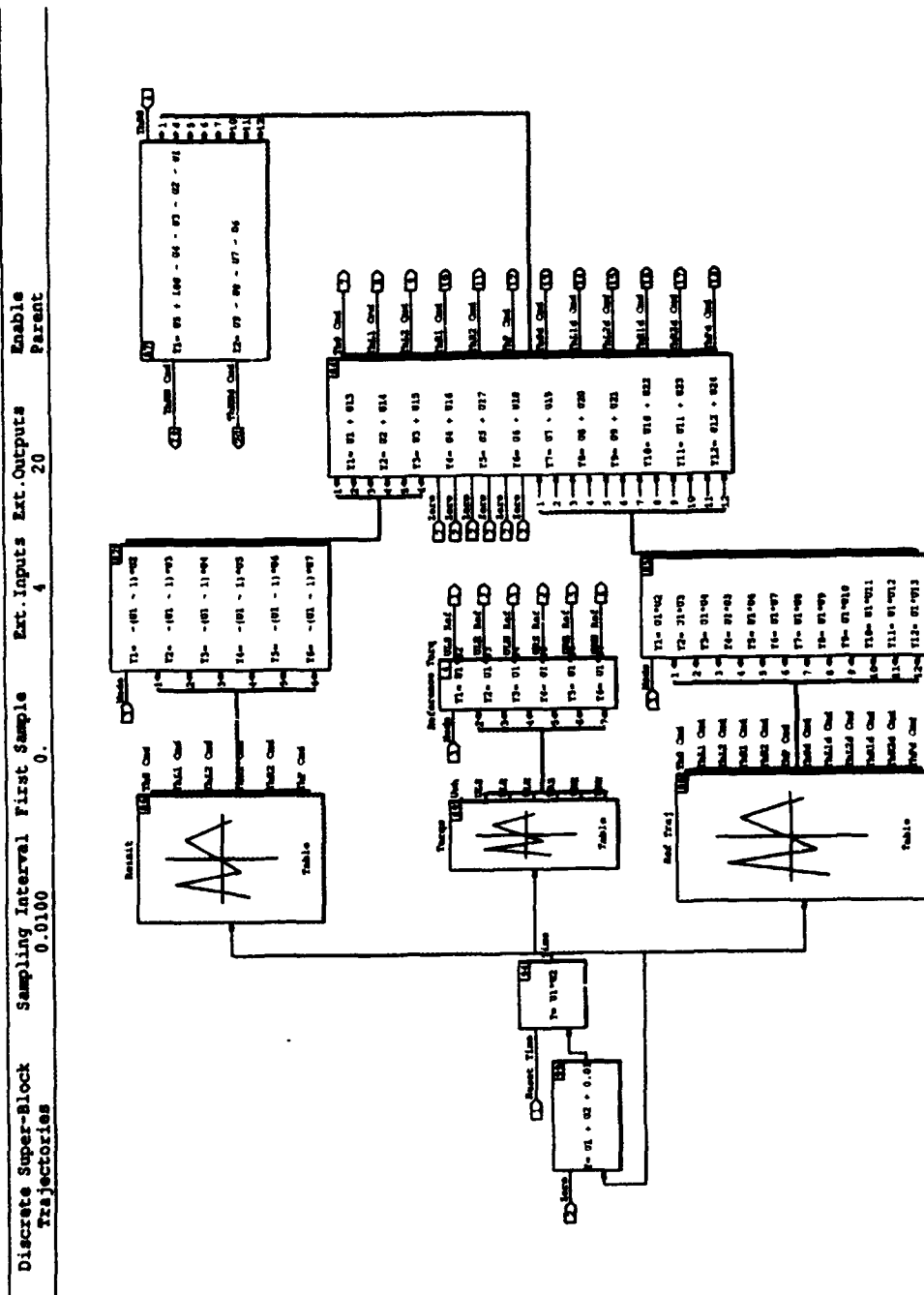


Figure 62: Reference Trajectory Block Diagram

27-JUL-93

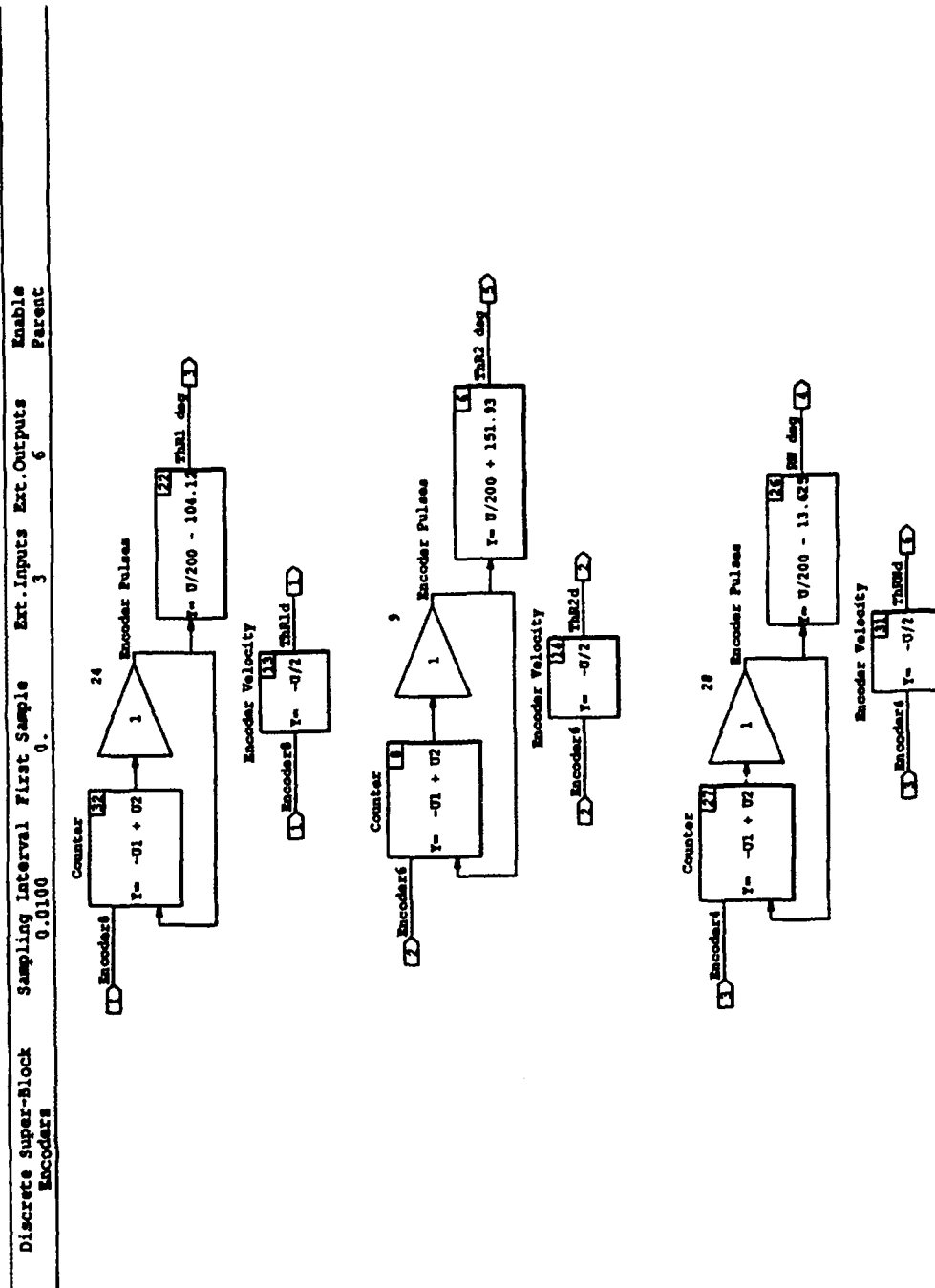


Figure 63: Encoders Block Diagram

27-JUL-93

Discrete Super-Block	Sampling Interval	First Sample	Ext. Inputs	Ext. Outputs	Enable
LeftAngles	0.0100	0.	18	6	Parent

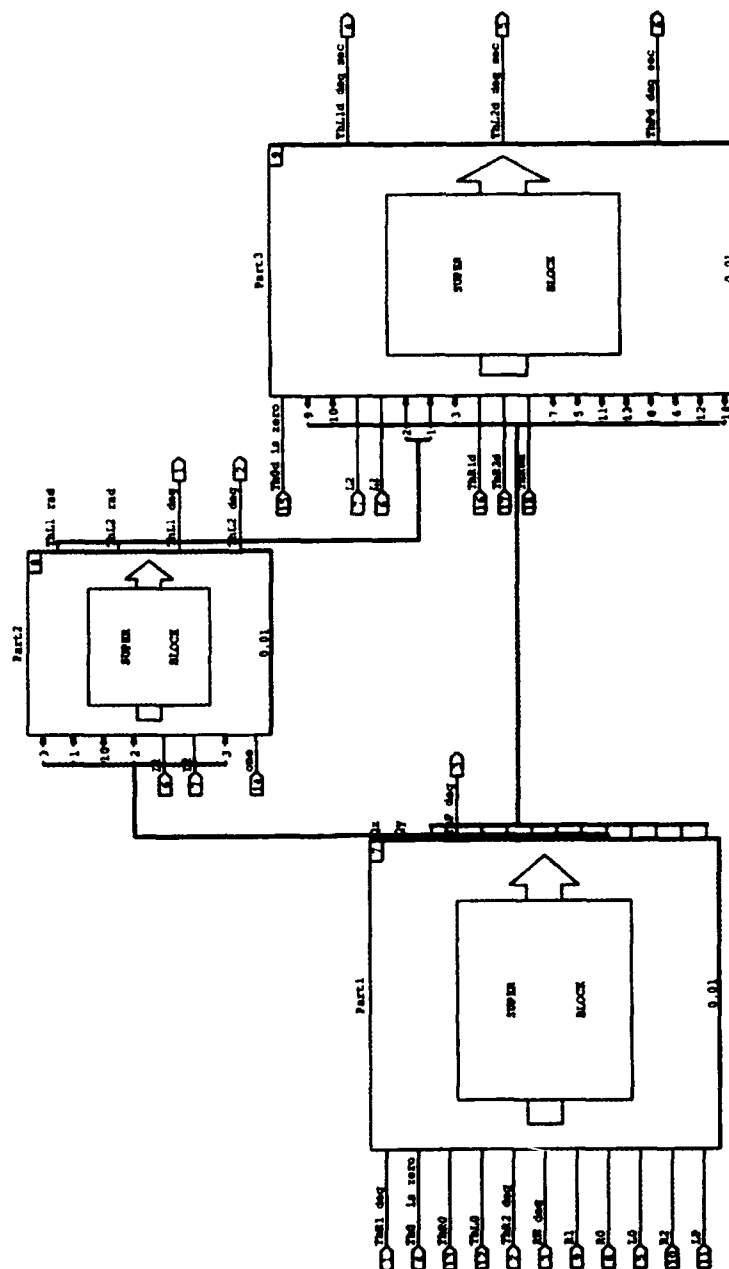


Figure 64: Left Angles Block Diagram

27-JUL-93

Discrete Super-Block	Sampling Interval	First Sample	Ext. Inputs	Ext. Outputs	Enable
Part 1	0.0100	0.	11	14	Parent

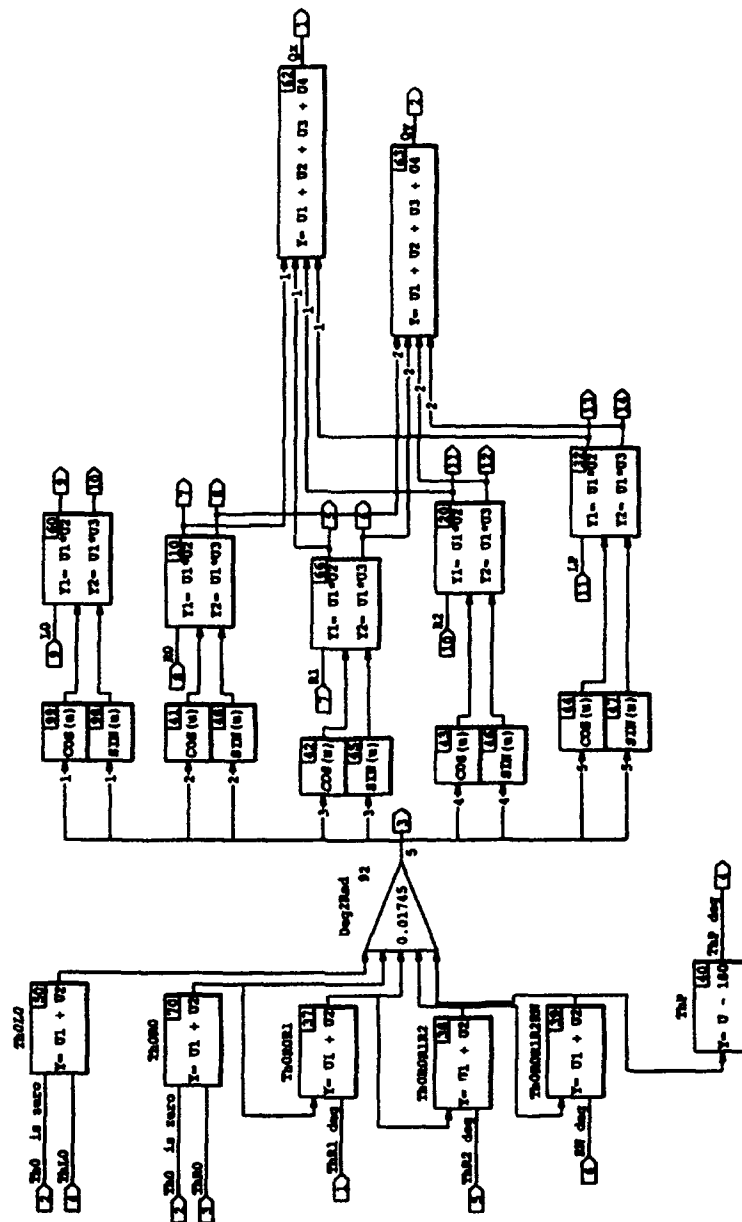
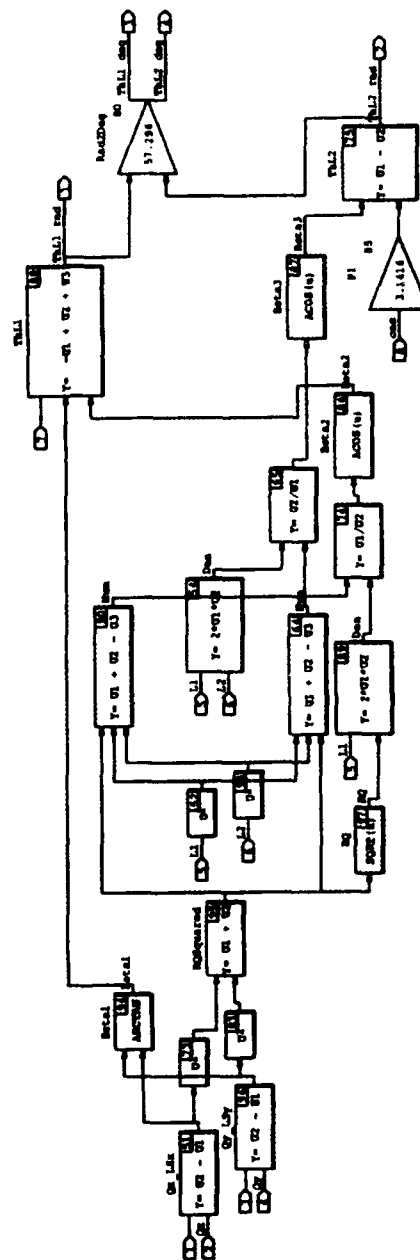
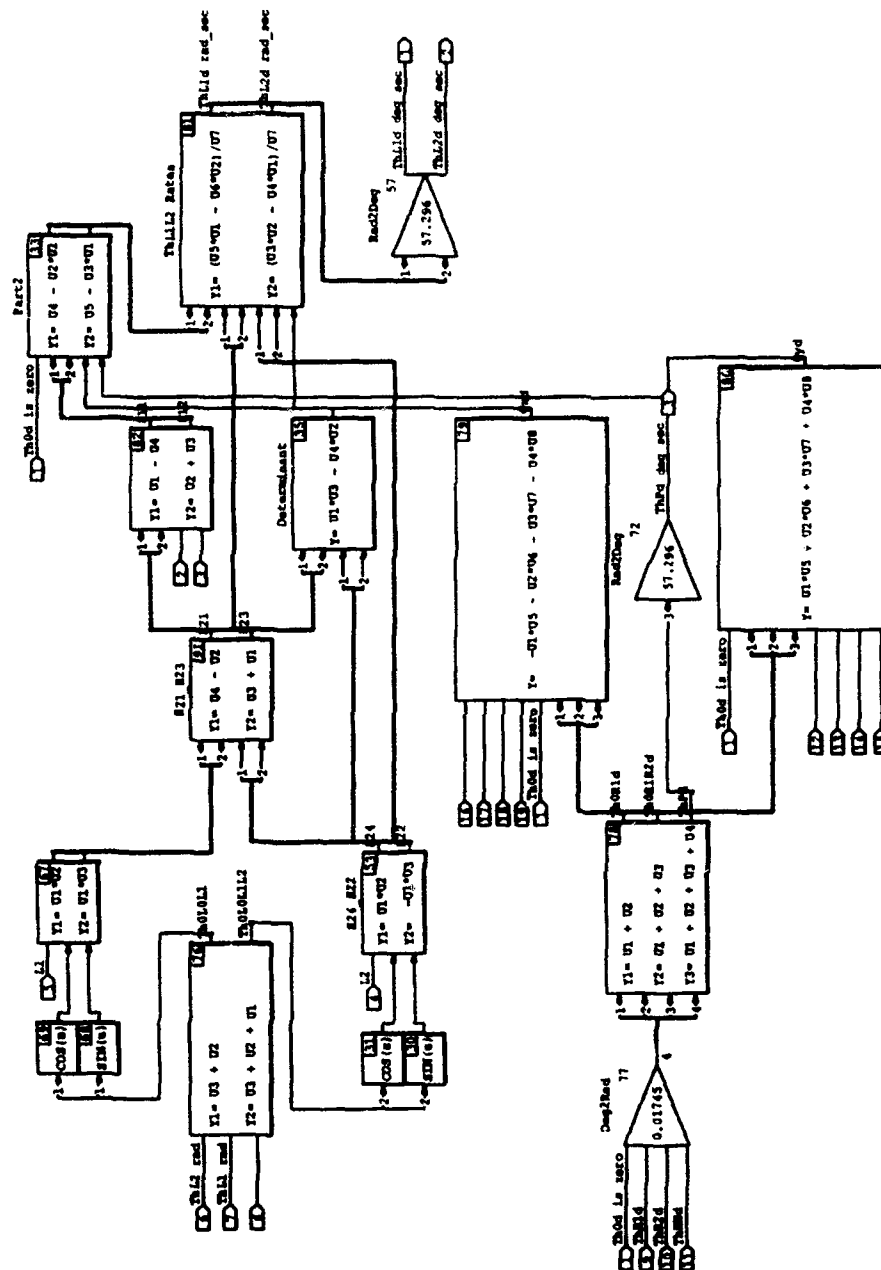


Figure 65: Part 1 Block Diagram

Discrete Super-Block	Sampling Interval	First Sample	Ext. Inputs	Ext. Outputs	Enable Parent
Part2	0.0100	0.	8	4	



105

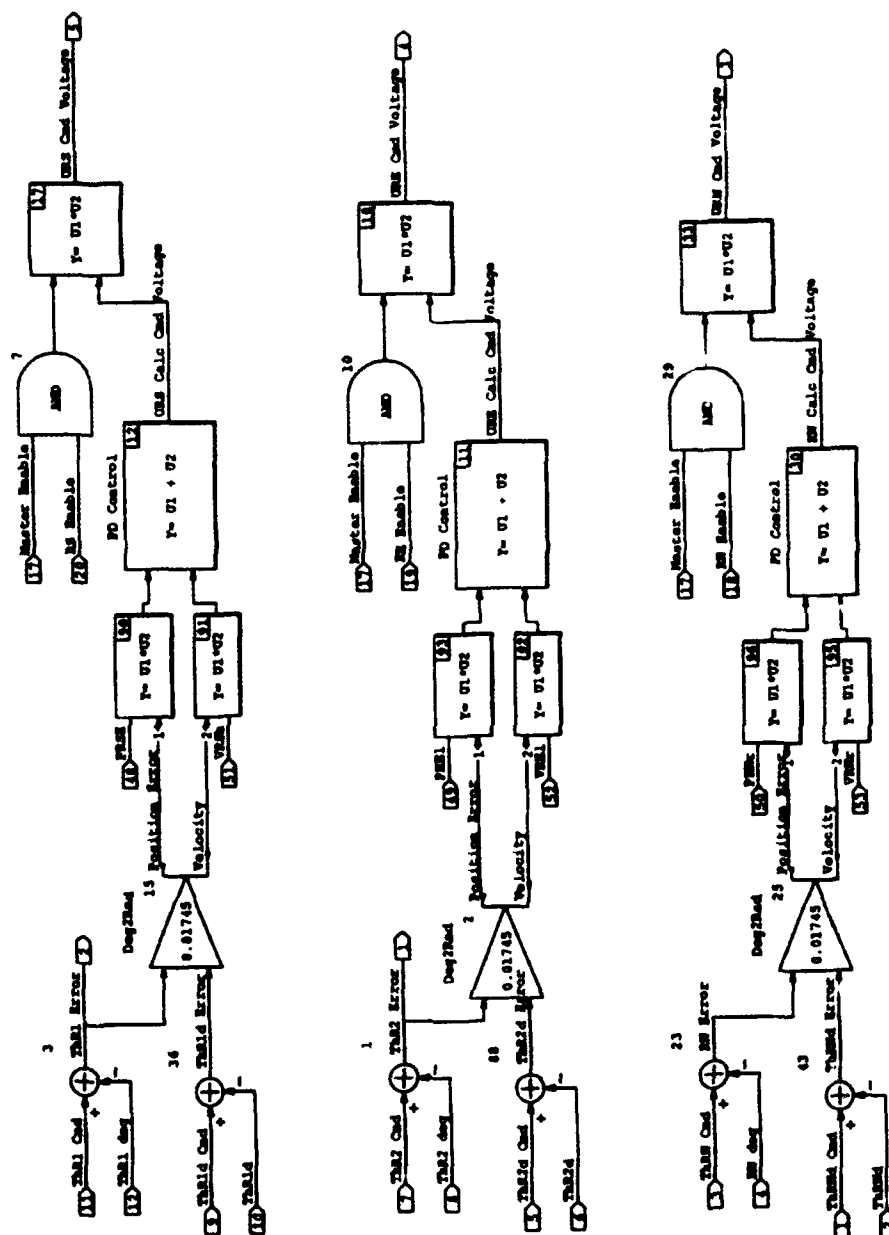


106



19-JUL-93

	Sampling Interval	First Sample	Ext. Inputs	Ext. Outputs	Enable Parent
Discrete Super-Block Controller	0.0100	0.	53	12	



**Figure 68: Right Manipulator Controller Block Diagram**

19-JUL-93

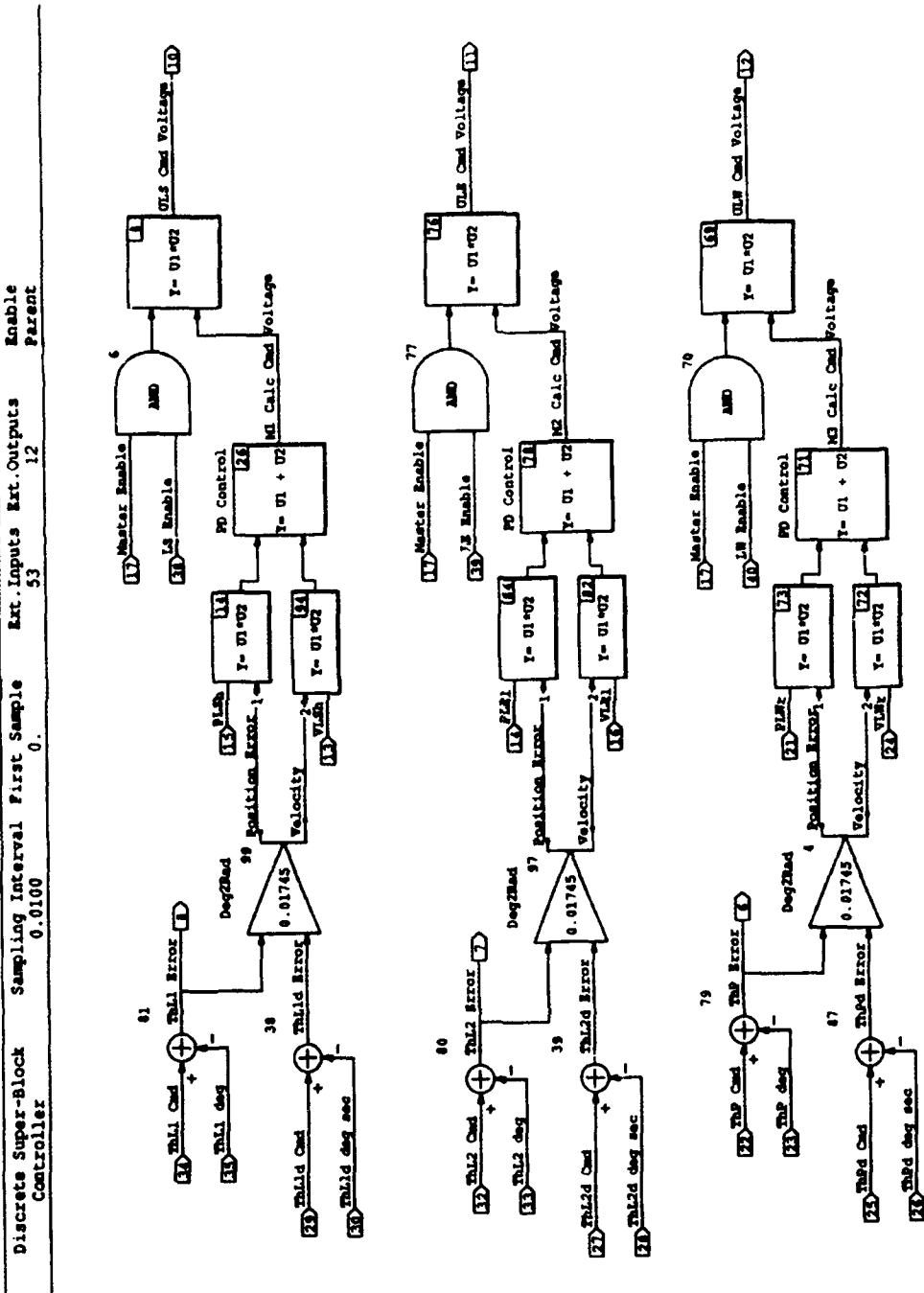


Figure 69: Left Manipulator Controller Block Diagram

## APPENDIX B: MATLAB CODE

The following listings are the MATLAB code used for the analytical simulations. The modules are included in alphabetical order. The hierarchical relationship between the modules is illustrated in Figure 70. The integration modules ode2 and odemin are minor variants of the MATLAB supplied module ode45. The modifications permit more parameters to be passed to and from these modules without having to include the extra variables in the state vector. Similarly, fminu2 modified the MATLAB unconstrained function minimization module, fminu, to include some diagnostic statements while running.

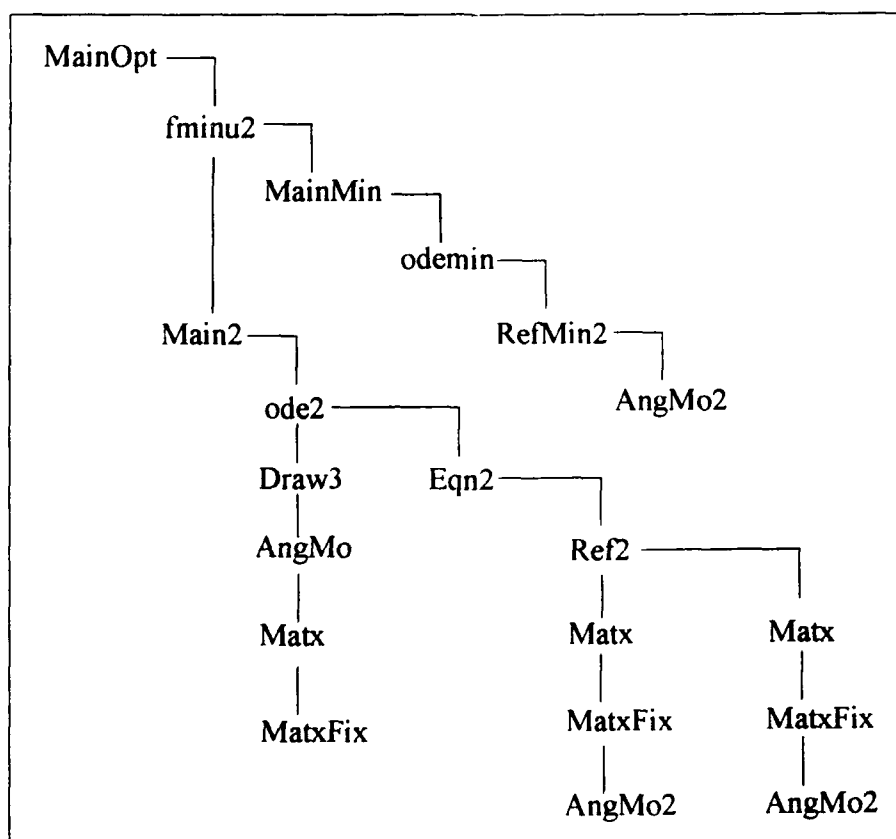


Figure 70: MATLAB Modules Hierarchy

## A. AngMo

```
% Filename is "AngMo.m"
% This file calculates the angular momentum of the system
function [Hs] = AngMo(Ls,Ms,CMs,Is,Q,Qdot)

% OUTPUT:
% Hs = 1x7 row vector of angular velocities. The first element is for
%       the centerbody. The next four elements are for the left upper
%       and lower arm followed by the right upper and lower arm. The
%       last two elements are for the payload and a total of all the
%       previous elements. [H0 HL1 HL2 HR1 HR2 HP HTotal]
%
% INPUT:
% Ls = 7x1 vector of lengths (m)
%       1st element = distance from origin to left arm mount
%       2nd & 3rd elements wrt left arm (from shoulder toward wrist)
%       4th element = payload length
%       5th & 6th elements wrt right arm (from wrist toward shoulder)
%       7th element = distance from right arm mount to origin
%       [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)
%       1st element = mass of spacecraft centerbody
%       2nd & 3rd elements = mass of left arm (upper arm then lower arm)
%       4th & 5th elements = mass of right arm (upper arm then lower arm)
%       6th element = payload mass
%       [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations
%       [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% Is = 6x1 column vector containing the moments of inertias about the
%       respective body's center of mass (kg m^2)
%       1st element = inertia of spacecraft centerbody
%       2nd & 3rd elements = inertia of left arm (upper arm then lower arm)
%       4th & 5th elements = inertia of right arm (upper arm then lower arm)
%       6th element = payload inertia
%       [I0; IL1; IL2; IR1; IR2; IP]
% Q = 8x1 column vector of generalized coordinates
% Qdot = 8x1 vector of generalized velocities

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lengths (m)
L0 = Ls(1);
L1 = Ls(2);
L2 = Ls(3);
LP = Ls(4);
R2 = Ls(5);
R1 = Ls(6);
R0 = Ls(7);

% Member masses (kg)
M0 = Ms(1);
ML1 = Ms(2);
ML2 = Ms(3);
MR1 = Ms(4);
MR2 = Ms(5);
MP = Ms(6);

% Center of mass distances (m)
```

```

Lc0 = CMs(1);
LcL1 = CMs(2);
LcL2 = CMs(3);
LcR1 = CMs(4);
LcR2 = CMs(5);
LcP = CMs(6); %measured from left end

% MOI about center of mass
I0 = Is(1);
IL1 = Is(2);
IL2 = Is(3);
IR1 = Is(4);
IR2 = Is(5);
IP = Is(6);

% Coordinates (rad & m)
Th0 = Q(1);
ThL1 = Q(2);
ThL2 = Q(3);
ThR1 = Q(4);
ThR2 = Q(5);
ThP = Q(6);
XP = Q(7);
YP = Q(8);

% Coordinate Rates (rad/sec & m/sec)
Th0d = Qdot(1);
ThL1d = Qdot(2);
ThL2d = Qdot(3);
ThR1d = Qdot(4);
ThR2d = Qdot(5);
ThPd = Qdot(6);
XPd = Qdot(7);
YPd = Qdot(8);

% Angular Momentum
H0 = Th0d*(I0 + M0*Lc0^2);
HL1 = Th0d*(IL1+ML1*(L0^2+LcL1^2+2*L0*LcL1*cos(ThL1))) + ...
    ThL1d*(IL1+ML1*(LcL1^2+L0*LcL1*cos(ThL1)));
HL2 = Th0d*(IL2+ML2*(L0^2+L1^2+LcL2^2+2*L0*L1*cos(ThL1) + ...
    2*L1*LcL2*cos(ThL2)+2*L0*LcL2*cos(ThL1+ThL2))) + ...
    ThL1d*(IL2+ML2*(L1^2+LcL2^2+L0*L1*cos(ThL1) + ...
    2*L1*LcL2*cos(ThL2)+L0*LcL2*cos(ThL1+ThL2))) + ...
    ThL2d*(IL2+ML2*(LcL2^2+L1*LcL2*cos(ThL2) + ...
    L0*LcL2*cos(ThL1+ThL2)));
HR1 = Th0d*(IR1+MR1*(R0^2+LcR1^2+2*R0*LcR1*cos(ThR1))) + ...
    ThR1d*(IR1+MR1*(LcR1^2+R0*LcR1*cos(ThR1)));
HR2 = Th0d*(IR2+MR2*(R0^2+R1^2+LcR2^2+2*R0*R1*cos(ThR1) + ...
    2*R1*LcR2*cos(ThR2)+2*R0*LcR2*cos(ThR1+ThR2))) + ...
    ThR1d*(IR2+MR2*(R1^2+LcR2^2+R0*R1*cos(ThR1) + ...
    2*R1*LcR2*cos(ThR2)+R0*LcR2*cos(ThR1+ThR2))) + ...
    ThR2d*(IR2+MR2*(LcR2^2+R1*LcR2*cos(ThR2) + ...
    R0*LcR2*cos(ThR1+ThR2)));
HP = ThPd*IP + MP*(-XPd*YP + YPd*XP);
HTotal = H0 + HL1 + HL2 + HR1 + HR2 + HP;
Hs = [H0 HL1 HL2 HR1 HR2 HP HTotal];

```

## B. AngMo2

```
% Filename is "AngMo2.m"
% This file calculates the angular momentum of the system
% Version 2 also finds the rate of change of angular momentum
function [Hs, Hdots] = AngMo2(Ls,Ms,CMs,Is,Q,Qdot,Qddot)

% OUTPUT:
% Hs = 1x7 row vector of angular velocities. The first element is for
%       the centerbody. The next four elements are for the left upper
%       and lower arm followed by the right upper and lower arm. The
%       last two elements are for the payload and a total of all the
%       previous elements. [H0 HL1 HL2 HR1 HR2 HP HTotal]
% Hdots = 1x7 row vector of the change in angular velocities. The order
%       is the same as for Hs
%
% INPUT:
% Ls = 7x1 vector of lengths (m)
%       1st element = distance from origin to left arm mount
%       2nd & 3rd elements wrt left arm (from shoulder toward wrist)
%       4th element = payload length
%       5th & 6th elements wrt right arm (from wrist toward shoulder)
%       7th element = distance from right arm mount to origin
%       [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)
%       1st element = mass of spacecraft centerbody
%       2nd & 3rd elements = mass of left arm (upper arm then lower arm)
%       4th & 5th elements = mass of right arm (upper arm then lower arm)
%       6th element = payload mass
%       [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations
%       [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% Is = 6x1 column vector containing the moments of inertias about the
%       respective body's center of mass (kg m^2)
%       1st element = inertia of spacecraft centerbody
%       2nd & 3rd elements = inertia of left arm (upper arm then lower arm)
%       4th & 5th elements = inertia of right arm (upper arm then lower arm)
%       6th element = payload inertia
%       [I0; IL1; IL2; IR1; IR2; IP]
% Q = 8x1 column vector of generalized coordinates
% Qdot = 8x1 vector of generalized velocities
% Qddot = 8x1 vector of generalized accelerations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lengths (m)
L0 = Ls(1);
L1 = Ls(2);
L2 = Ls(3);
LP = Ls(4);
R2 = Ls(5);
R1 = Ls(6);
R0 = Ls(7);

% Member masses (kg)
M0 = Ms(1);
ML1 = Ms(2);
ML2 = Ms(3);
MR1 = Ms(4);
```

```

MR2 = Ms(5);
MP = Ms(6);

% Center of mass distances (m)
Lc0 = CMs(1);
LcL1 = CMs(2);
LcL2 = CMs(3);
LcR1 = CMs(4);
LcR2 = CMs(5);
LcP = CMs(6); %measured from left end

% MOI about center of mass
I0 = Is(1);
IL1 = Is(2);
IL2 = Is(3);
IR1 = Is(4);
IR2 = Is(5);
IP = Is(6);

% Coordinates (rad & m)
Th0 = Q(1);
ThL1 = Q(2);
ThL2 = Q(3);
ThR1 = Q(4);
ThR2 = Q(5);
ThP = Q(6);
XP = Q(7);
YP = Q(8);

% Coordinate Rates (rad/sec & m/sec)
Th0d = Qdot(1);
ThL1d = Qdot(2);
ThL2d = Qdot(3);
ThR1d = Qdot(4);
ThR2d = Qdot(5);
ThPd = Qdot(6);
XPd = Qdot(7);
YPd = Qdot(8);

% Coordinate Accelerations (rad/sec^2 & m/sec^2)
Th0dd = Qddot(1);
ThL1dd = Qddot(2);
ThL2dd = Qddot(3);
ThR1dd = Qddot(4);
ThR2dd = Qddot(5);
ThPdd = Qddot(6);
XPdd = Qddot(7);
YPdd = Qddot(8);

% Angular Momentum
H0 = Th0d*(I0 + M0*Lc0^2);
HL1 = Th0d*(IL1+ML1*(L0^2+LcL1^2+2*L0*LcL1*cos(ThL1))) + ...
      ThL1d*(IL1+ML1*(LcL1^2+L0*LcL1*cos(ThL1)));
HL2 = Th0d*(IL2+ML2*(L0^2+L1^2+LcL2^2+2*L0*L1*cos(ThL1) + ...
      2*L1*LcL2*cos(ThL2)+2*L0*LcL2*cos(ThL1+ThL2))) + ...
      ThL1d*(IL2+ML2*(L1^2+LcL2^2+L0*L1*cos(ThL1) + ...
      2*L1*LcL2*cos(ThL2)+L0*LcL2*cos(ThL1+ThL2))) + ...
      ThL2d*(IL2+ML2*(LcL2^2+L1*LcL2*cos(ThL2) + ...
      L0*LcL2*cos(ThL1+ThL2)));
HR1 = Th0d*(IR1+MR1*(R0^2+LcR1^2+2*R0*LcR1*cos(ThR1))) + ...
      ThR1d*(IR1+MR1*(LcR1^2+R0*LcR1*cos(ThR1)));

```

```

IR2 = Th0d*(IR2+MR2*(R0^2+R1^2+LcR2^2+2*R0*R1*cos(ThR1) + ...
      2*R1*LcR2*cos(ThR2)+2*R0*LcR2*cos(ThR1+ThR2))) + ...
      ThR1d*(IR2+MR2*(R1^2+LcR2^2+R0*R1*cos(ThR1) + ...
      2*R1*LcR2*cos(ThR2)+R0*LcR2*cos(ThR1+ThR2))) + ...
      ThR2d*(IR2+MR2*(LcR2^2+R1*LcR2*cos(ThR2) + ...
      R0*LcR2*cos(ThR1+ThR2)));
IP = ThPd*IP + MP*(-XPd*YP + YPd*XP);
ITotal = H0 + HL1 + IIL2 + HR1 + HR2 + HP;
IIs = [H0 HL1 HL2 IIR1 IIR2 IIP ITotal];

% Change in angular momentum
H0d = Th0dd*(I0 + M0*Lc0^2);
IIL1d = Th0dd*(IL1+ML1*(L0^2+LcL1^2+2*L0*LcL1*cos(ThL1))) + ...
      ThL1dd*(IL1+ML1*(LcL1^2+L0*LcL1*cos(ThL1))) - ...
      Th0d*ThL1d*2*ML1*L0*LcL1*sin(ThL1) - ...
      ThL1d^2*ML1*L0*LcL1*sin(ThL1);
IIL2d = Th0dd*(IL2+ML2*(L0^2+L1^2+LcL2^2+2*L0*L1*cos(ThL1) + ...
      2*L1*LcL2*cos(ThL2)+2*L0*LcL2*cos(ThL1+ThL2))) + ...
      ThL1dd*(IL2+ML2*(L1^2+LcL2^2+L0*L1*cos(ThL1) + ...
      2*L1*LcL2*cos(ThL2)+L0*LcL2*cos(ThL1+ThL2))) + ...
      ThL2dd*(IL2+ML2*(LcL2^2+L1*LcL2*cos(ThL2) + ...
      L0*LcL2*cos(ThL1+ThL2))) - ...
      Th0d*ThL1d*2*ML2*(L0*L1*sin(ThL1)+L0*LcL2*sin(ThL1+ThL2)) - ...
      Th0d*ThL2d*2*ML2*(L1*LcL2*sin(ThL2)+L0*LcL2*sin(ThL1+ThL2)) - ...
      ThL1d*ThL2d*2*ML2*(L1*LcL2*sin(ThL2)+L0*LcL2*sin(ThL1+ThL2))-...
      ThL1d^2*ML2*(L0*L1*sin(ThL1)+L0*LcL2*sin(ThL1+ThL2)) - ...
      ThL2d^2*ML2*(L1*LcL2*sin(ThL2)+L0*LcL2*sin(ThL1+ThL2));
IIR1d = Th0dd*(IR1+MR1*(R0^2+LcR1^2+2*R0*LcR1*cos(ThR1))) + ...
      ThR1dd*(IR1+MR1*(LcR1^2+R0*LcR1*cos(ThR1))) - ...
      Th0d*ThR1d*2*MR1*R0*LcR1*sin(ThR1) - ...
      ThR1d^2*MR1*R0*LcR1*sin(ThR1);
HR2d = Th0dd*(IR2+MR2*(R0^2+R1^2+LcR2^2+2*R0*R1*cos(ThR1) + ...
      2*R1*LcR2*cos(ThR2)+2*R0*LcR2*cos(ThR1+ThR2))) + ...
      ThR1dd*(IR2+MR2*(R1^2+LcR2^2+R0*R1*cos(ThR1) + ...
      2*R1*LcR2*cos(ThR2)+R0*LcR2*cos(ThR1+ThR2))) + ...
      ThR2dd*(IR2+MR2*(LcR2^2+R1*LcR2*cos(ThR2) + ...
      R0*LcR2*cos(ThR1+ThR2))) - ...
      Th0d*ThR1d*2*MR2*(R0*R1*sin(ThR1)+R0*LcR2*sin(ThR1+ThR2)) - ...
      Th0d*ThR2d*2*MR2*(R1*LcR2*sin(ThR2)+R0*LcR2*sin(ThR1+ThR2))-...
      ThR1d*ThR2d*2*MR2*(R1*LcR2*sin(ThR2)+R0*LcR2*sin(ThR1+ThR2))-...
      ThR1d^2*MR2*(R0*R1*sin(ThR1)+R0*LcR2*sin(ThR1+ThR2)) - ...
      ThR2d^2*MR2*(R1*LcR2*sin(ThR2)+R0*LcR2*sin(ThR1+ThR2));
HPd = ThPdd*IP + MP*(-XPdd*YP - XPd*YPd + YPd*XP + YPd*XPd);
HdTotal = H0d + HL1d + HL2d + HR1d + HR2d + HPd;
Hdots = [H0d HL1d HL2d HR1d HR2d HPd HdTotal];

```

### C. Draw3

% Filename is 'Draw3.m'

```
function[X,Y] = Draw3(Lengths,AngConst,AngHist,Interval)
```

```

%%%%%%%%%%
%
% This file draws the dual arm spacecraft stick figure
%
% INPUTS:
%
% Lengths = 7x1 vector of link lengths (m)

```



```

%      1st element is distance from origin to left arm mount
%      2nd & 3rd elements wrt left arm (from shoulder toward wrist)
%      4th element is payload length
%      5th & 6th elements wrt right arm (from wrist toward shoulder)
%      7th element is distance from right arm mount to origin
% AngConst = vector of angles to arm mounting locations wrt centerbody coord
%            frame (angle for left arm then angle for right arm)
% AngHist = nx6 matrix of angle histories. Each row represents a
%           specific time. Each column represents a specific joint
%           angle (except the payload angle is inertial)
%           1st column is the center body angle
%           2nd & 3rd columns are the left arm shoulder and elbow
%           4th & 5th columns are the right arm shoulder and elbow
%           6th column is the payload (this angle is inertial)
% Interval = plot every "interval"th time
%
%
% OUTPUTS:
%
% X = vector history of joint x coordinates
% Y = vector history of joint y coordinates
% X & Y treat the system as a closed chain beginning at the centerbody origin,
% outward along the left arm, across the payload, inward along the right arm,
% and back to the origin.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Times,dummy] = size(AngHist);
Links = length(Lengths);
X(1,1) = 0;
Y(1,1) = 0;
% Convert the joint angles to inertial angles and reorder them for closed chain use
NAng(:,1) = AngHist(:,1) + AngConst(1)*ones(Times,1);
NAng(:,2) = NAng(:,1) + AngHist(:,2);
NAng(:,3) = NAng(:,2) + AngHist(:,3);
NAng(:,4) = NAng(:,3) + AngHist(:,4);
NAng(:,5) = NAng(:,4) + AngHist(:,5);
NAng(:,6) = NAng(:,5) + AngHist(:,6);
NAng(:,7) = NAng(:,6) + AngConst(2)*ones(Times,1) + pi;
NAng(:,8) = NAng(:,7) + AngHist(:,7);
NAng(:,9) = NAng(:,8) + AngHist(:,8);

p = 1;
while p <= Times
    for q = 1:Links
        Lastx = 0;
        Lasty = 0;
        for r = 1:9
            Lastx = Lastx + Lengths(r)*cos(NAng(p,r));
            Lasty = Lasty + Lengths(r)*sin(NAng(p,r));
        end
        X(q+1,p) = Lastx;
        Y(q+1,p) = Lasty;
    end
    p = p + Interval;
end
X = [X(1:Links,:); X(2,:); X(Links,:); X(Links+1,:)];
Y = [Y(1:Links,:); Y(2,:); Y(Links,:); Y(Links+1,:)];

% Plot the Final Case
for q = 1:Links
    Lastx = 0;
    Lasty = 0;

```

```

for r = 1:q
    Lastx = Lastx + Lengths(r)*cos(NAng(Times,r));
    Lasty = Lasty + Lengths(r)*sin(NAng(Times,r));
end
XFinal(q+1,1) = Lastx;
YFinal(q+1,1) = Lasty;
end
XFinal = [XFinal(1:Links,:); XFinal(2,:); XFinal(Links,:); XFinal(Links+1,:)];
YFinal = [YFinal(1:Links,:); YFinal(2,:); YFinal(Links,:); YFinal(Links+1,:)];

clg;
axis('square')
plot(X,Y,'-',XFinal,YFinal,'-',XFinal,YFinal,'x', X(1),Y(1),'o');
xlabel('X (m)'),ylabel('Y (m)');
axis('normal')

```

#### D. Eqn2

```

% Filename is 'Eqn2.m'
% Differential Equations for numerical integrator
function [Xdots,U,TorqRef,Aqdot,J,Res,LHS,RHS,Delq] = ...
    Eqn2(T,X,Ls,Ms,CMs,ls,BoundC,Gains,XfDes,Wu,Wc,Coef,ConstMat)

% OUTPUT:
% xdot = derivatives of state vector at time T
% U = column vector of actual torques commanded at time T arranged
%     as [U1; U2; U6; U5] where the number denotes the joint
%     associated with that torque
% TorqRef = column vector of reference torques that should be applied
%           at time T if the motion followed the reference maneuver exactly.
%           These are arranged in the same order as U.
% Res = column vector of residuals after EOM are evaluated with the
%       calculated reference torques. (Residuals should be zero).
% Aqdot = column vector of A*qdot. This is a test to see if the
%          constraint equation (A*qdot = 0) is satisfied.
% LHS = column vector of the EOM left hand side (LHS = M*qddot + GTilda)
% RHS = column vector of the EOM right hand side (RHS = BTilda*u)
%
% INPUTS:
% T = time (sec)
% State Vector, X, is defined as follows:
% X1 = Theta 0 (rad)
% X2 = Theta L1 (rad)
% X3 = Theta L2 (rad)
% X4 = Theta R1 (rad)
% X5 = Theta R2 (rad)
% X6 = Theta P (rad)
% X7 = X component of Payload Center of mass position (m)
% X8 = Y component of Payload Center of mass position (m)
% X9 = Theta 0 Dot (rad/sec)
% X10 = Theta L1 Dot (rad/sec)
% X11 = Theta L2 Dot (rad/sec)
% X12 = Theta R1 Dot (rad/sec)
% X13 = Theta R2 Dot (rad/sec)
% X14 = Theta P Dot (rad/sec)
% X15 = X component of Payload Center of mass velocity (m/sec)
% X16 = Y component of Payload Center of mass velocity (m/sec)
% X17 = integral of the absolute value of the centerbody disturbance torque
% X18 = integral of the centerbody disturbance torque squared

```

```

% Ls = 7x1 vector of lengths (m)
%   1st element = distance from origin to left arm mount
%   2nd & 3rd elements wrt left arm (from shoulder toward wrist)
%   4th element = payload length
%   5th & 6th elements wrt right arm (from wrist toward shoulder)
%   7th element = distance from right arm mount to origin
%   [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)
%   1st element = mass of spacecraft centerbody
%   2nd & 3rd elements = mass of left arm (upper arm then lower arm)
%   4th & 5th elements = mass of right arm (upper arm then lower arm)
%   6th element = payload mass
%   [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations
%   [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% Is = 6x1 column vector containing the moments of inertias about the
%   respective body's center of mass (kg m^2)
%   1st element = inertia of spacecraft centerbody
%   2nd & 3rd elements = inertia of left arm (upper arm then lower arm)
%   4th & 5th elements = inertia of right arm (upper arm then lower arm)
%   6th element = payload inertia
%   [I0; IL1; IL2; IR1; IR2; IP]
% BoundC = boundry conditions for the problem. The first column
%   contains the initial x and y component of points Q & P
%   respectively, the x component of the right arm base, the
%   problem start time, and the simulation stop time. The second
%   column contains the x and y component of points Q & P
%   respectively, the x component of the right arm base, the
%   stop time for the ideal reference maneuver, and a flag to
%   activate or deactivate the controller. The origin for the
%   x and y components is the base of the left arm.
% Wu = 6x6 control torque cost weighting matrix
% Wc = 8x8 constraint cost weighting matrix
% Gains = 1x2 column vector of controller gains. The first value is
%   for position gains and the second value is for velocity
%   gains.
% XfDes = column vector containing desired values for the angles at
%   the conclusion of the maneuver. These are the same angles
%   the reference maneuver is trying to create. They are arranged
%   as [Th0f; ThL1f; ThL2f; ThR1f; ThR2f; ThPf].

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Th0 = X(1);
ThL1 = X(2);
ThL2 = X(3);
ThR1 = X(4);
ThR2 = X(5);
ThP = X(6);
Xc = X(7);
Yc = X(8);
Th0d = X(9);
ThL1d = X(10);
ThL2d = X(11);
ThR1d = X(12);
ThR2d = X(13);
ThPd = X(14);
Xcd = X(15);
Ycd = X(16);

```

```

% Arms mount locations wrt spacecraft centerbody coordinate frame (rad)
ThL0 = BoundC(5,1); ThR0 = BoundC(5,2);

% Stop Times
TfR = BoundC(6,2); % Reference Torque stop time (sec)
TfC = BoundC(7,1); % Controller stop time (sec)

% Controller Flag
ContFlag = BoundC(7,2);

% Constraints Matrix Flag
AMatFlag = BoundC(8,1);

% Centerbody Reaction Wheel Flag
WheelFlag = BoundC(8,2);

% Kinetic Energy Test Flag
KEFlag = BoundC(11,1);

% Inverse Kinematics Bypass Flag
ByPass = BoundC(11,2);

% Torque selection if bypass is enabled
TorqFlag = BoundC(12,1);

% Maximum torque from reaction wheel
TorqCap = BoundC(13,1); % Limit enabled
TorqMax = BoundC(13,2); % Limit amount

% Controller Gains:
Gpos = Gains(1);
Gvel = Gains(2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATIONS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EOM:  $M \cdot \ddot{q} + dV/d\dot{q} + G = Q_f + A' \cdot \Lambda$ 
% M is mass matrix
%  $\ddot{q}$  is column vector of generalized coordinate accelerations
%  $dV/d\dot{q}$  is the partial derivative of the potential function with
% respect to the generalized coordinates. This term is zero for
% this problem because all motion is in the horizontal plane (there
% is no change in potential energy caused by the motion)
% G is a column vector which is a function of q and  $\dot{q}$ 
%  $Q_f$  are generalized forces caused by joint torques
% A' is transpose of constraints matrix
%  $\Lambda$  are Lagrange multipliers

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% State Vector & Derivative %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Q = [Th0; ThL1; ThL2; ThR1; ThR2; ThP; Xc; Yc];
Qdot = [Th0d; ThL1d; ThL2d; ThR1d; ThR2d; ThPd; Xcd; Ycd];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Matrices %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
AngConst = [ThL0; ThR0];
if AMatFlag
    [M,G,A,Adot,B] = MatxFix(Ls,Ms,CMs,Is,Q,Qdot,AngConst);
else

```

```

[M,G,A,Adot,B] = Matx(Ls,Ms,CMs,ls,Q,Qdot,AngConst);
end

if WheelFlag
    B7 = [1; 0; 0; 0; 0; 0; 0; 0];
    B = [B7 B];
end

if ByPass % If true, then bypass calculating torques using inverse
          % kinematics. This branch of logic is a verification test
          % during program development and is not intended for regular
          % use once the program is checked out.
    if TorqFlag == 0
        U = zeros(6,1); J = 0;
    else
        if TorqFlag == 1
            U = [-0.01; 0; 0; 0; 0; 0]; J = 0;
        else
            U = [-0.01; 0; 0; 0.01; 0; 0]; J = 0;
        end
    end
    if WheelFlag
        U = [0; U];
    end
else % Normal program flow to find control torques
    %%%%%%%%%%%
    %%% Torques %%%
    %%%%%%%%%%%
    if T <= TfR, % Get the appropriate torque and angle values
                  % from the reference maneuver calculations
        [TorqRef, QRef, QdotRef, Aqdot, J, C1Ref, C2Ref, C3Ref] = ...
            Ref2(Ls,Ms,CMs,ls,BoundC,T,Wu,Wc,Coef,ConstMat);
    else % Simulation is longer than ideal reference maneuver
        % Use no reference torques
        % Use the desired final values as references
        TorqRef = [0; 0; 0; 0; 0; 0];
        QRef(1) = XfDes(1);
        QRef(2) = XfDes(2);
        QRef(3) = XfDes(3);
        QRef(4) = XfDes(4);
        QRef(5) = XfDes(5);
        QRef(6) = XfDes(6);
        QRef(7) = XfDes(7);
        QRef(8) = XfDes(8);
        QdotRef(1) = XfDes(9);
        QdotRef(2) = XfDes(10);
        QdotRef(3) = XfDes(11);
        QdotRef(4) = XfDes(12);
        QdotRef(5) = XfDes(13);
        QdotRef(6) = XfDes(14);
        QdotRef(7) = XfDes(15);
        QdotRef(8) = XfDes(16);
        if WheelFlag
            TorqRef = [0; TorqRef];
        end

        % Matrices
        if AMatFlag
            [MRef,GRef,ARef,AdotRef] = MatxFix(Ls,Ms,CMs,ls,...
                QRef,QdotRef,AngConst);
        else

```

```

[MRef,GRef,ARef,AdotRef] = Matx(Ls,Ms,CMs,Is,QRef,QdotRef,...
                                AngConst);
end
BRef = B;
Pt1Ref = ARef*inv(ARef*inv(MRef)*ARef');
C1Ref = inv(MRef)*(eye(M) - Pt1Ref*ARef*inv(MRef))*BRef;
C2Ref = -inv(MRef)*Pt1Ref*AdotRef;
C3Ref = inv(MRef)*(Pt1Ref*ARef*inv(MRef) - eye(M))*GRef;
end

if ContFlag % Controller is on
    Delq = Q - QRef;
    Delqdot = Qdot - QdotRef;
    % Controller calculations
    Pt1 = A'*inv(A*inv(M)*A');
    C1 = inv(M)*(eye(M) - Pt1*A*inv(M))*B;
    C2 = -inv(M)*Pt1*Adot;
    C3 = inv(M)*(Pt1*A*inv(M) - eye(M))*G;
    F2 = Gpos * Delq;
    F2 = [F2(1:6); 0; 0];
    Kv = Gvel * eye(M);
    Kv(7,7)=0; Kv(8,8)=0;
    Pt3 = pinv(C1);
    % Pt3 = inv(C1'*C1)*C1'; % Resulted in poorly conditioned matrix

    %%%%%%%%%%%%%%%
    % Complete Lyapunov Controller %
    %%%%%%%%%%%%%%%
    U = Pt3*(-Kv*Delqdot + C1Ref*TorqRef - (C2*Qdot - C2Ref*QdotRef) - ...
            (C3 - C3Ref) - F2);
    %%%%%%%%%%%%%%%
    % Simplified Lyapunov Controller %
    % (removes reference torques and %
    % assumes C2 and C3 terms are small) %
    %%%%%%%%%%%%%%%
    % Kp = Gpos * eye(M);
    % Pt3 = pinv(C1Ref);
    % U = Pt3*(-(Kv+C2Ref)*Delqdot - Kp*Delq) + TorqRef;

else % Controller is off
    U = TorqRef; % Don't adjust torques from reference maneuver
    Delq = 999*ones(8,1); % Dummy value for trajectory error
end % End of Control Loop

if WheelFlag
    J = abs(U(1));
else
    J = 0;
end
end % End of Kinetic Energy Test Conditional

if TorqCap % Upper limit on wheel torque enabled?
    if abs(U(1)) > TorqMax
        if U(1) > 0
            U(1) = TorqMax;
        else
            U(1) = -TorqMax;
        end
    end
end
end

%%%%%%%%%%%%%%

```

```

%% Qf %%
%% Lagrange Multipliers %%
% Qf = B*u These are the generalized forces
Qf = B * U;

%% EOM: M*qddot + dV/dq + G = Qf + A'*Lam
% Solving the EOM for qddot gives: qddot = inv(M)*(Qf + A'*Lam - G)
% Differentiating the Pfaffian form of the constraint equations
% results in: Aidot*qdot + A*qddot = 0.
% Substitution of the expression for qddot into the previous equation
% permits solving for Lam:
% Lam = inv(A*inv(M)*A')*(A*inv(M)*(G-Qf) - Aidot*qdot);
Lam = inv(A*inv(M)*A')*(A*inv(M)*(G-Qf) - Aidot*qdot);

%% Putting it all together %%
Qddot = inv(M) * (Qf + A'*Lam - G);

[His, Hdots] = AngMo2(Ls,Ms,CMs,Is,Q,Qdot,Qddot);
% Change in total angular momentum
HId = Hdots(7);
J = [J; Hd];

% Assemble derivative of state vector for integrator
Xdot = [Qdot; Qddot; J(1); (J(1))^2];

%% Troubleshooting Info %%
Aqdot = A*Qdot;
LIIS = M*Qddot + G;
RHIS = Qf + A'*Lam;
Res = LIIS - RHIS;

```

## E. fminu2

```

function [x,OPTIONS] = fminu2(FUN,x,OPTIONS,GRADFUN,P1,P2,P3,P4,P5,P6,...
    P7,P8,P9,P10)
%FMINU Finds the minimum of a function of several variables.
% X=FMINU('FUN',X0) starts at the matrix X0 and finds a minimum to the
% function which is described in FUN (usually an M-file: FUN.M).
% The function 'FUN' should return a scalar function value: F=FUN(X).
%
% X=FMINU('FUN',X0,OPTIONS) allows a vector of optional parameters to
% be defined. OPTIONS(1) controls how much display output is given; set
% to 1 for a tabular display of results, (default is no display: 0).
% OPTIONS(2) is a measure of the precision required for the values of
% X at the solution. OPTIONS(3) is a measure of the precision
% required of the objective function at the solution.
% For more information type HELP FOPTIONS.
%
% X=FMINU('FUN',X0,OPTIONS,'GRADFUN') enables a function'GRADFUN'
% to be entered which returns the partial derivatives of the function,
% df/dX, at the point X: gf = GRADFUN(X).
%
```

```
% The default algorithm is the BFGS Quasi-Newton method with a
% mixed quadratic and cubic line search procedure.
```

```
% Copyright (c) 1990 by the MathWorks, Inc.
% Andy Grace 7-9-90.
```

```
% -----[Initialization]-----
```

```
XOUT=x(:);
nvars=length(XOUT);
```

```
evalstr = [FUN];
if ~any(FUN<48)
evalstr=[evalstr, 'x'];
for i=1:nargin - 4
    evalstr = [evalstr, 'P', num2str(i)];
end
evalstr = [evalstr, ')'];
end
```

```
if nargin < 3, OPTIONS=[]; end
if nargin < 4, GRADFUN=[]; end
```

```
if length(GRADFUN)
evalstr2 = [GRADFUN];
if ~any(GRADFUN<48)
    evalstr2 = [evalstr2, 'x'];
    for i=1:nargin - 4
        evalstr2 = [evalstr2, 'P', num2str(i)];
    end
    evalstr2 = [evalstr2, ')'];
end
end
```

```
f = eval(evalstr);
n = length(XOUT);
GRAD=zeros(nvars,1);
OLDX=XOUT;
MATX=zeros(3,1);
MATL=[f;0;0];
OLDF=f;
FIRSTF=f;
[OLDX,OLDF,HESS,OPTIONS]=optint(XOUT,f,OPTIONS);
CHG = 1e-7*abs(XOUT)+1e-7*ones(nvars,1);
SD = zeros(nvars,1);
diff = zeros(nvars,1);
```

```
OPTIONS(10)=2; % Iteration count (add 1 for last evaluation)
status =-1;
```

```
while status ~= 1
% Work Out Gradients
if ~length(GRADFUN) | OPTIONS(9)
    OLDF=f;
% Finite difference perturbation levels
% First check perturbation level is not less than search direction.
    f = find(10*abs(CHG)>abs(SD));
    CHG(f) = -0.1*SD(f);
% Ensure within user-defined limits
```



```

        CHG = sign(CHG+eps).*min(max(abs(CHG),OPTIONS(16)),OPTIONS(17));
        for gcnt=1:nvars
            XOUT(gcnt,1)=XOUT(gcnt)+CHG(gcnt);
            OPTIONS(10)=OPTIONS(10)+1;
            disp('While Loop Iteration in Progress');
            disp(['Iterations: ',num2str(OPTIONS(10))]);
            disp(['Allowable: ',num2str(OPTIONS(14))]);
            x(:) = XOUT; f = eval(evalstr);
            GRAD(gcnt)=(f-OLDF)/(CHG(gcnt));
            if f < OLDF
                OLDF=f;
            else
                XOUT(gcnt)=XOUT(gcnt)-CHG(gcnt);
            end
        end
        % Try to set difference to 1e-8 for next iteration
        CHG = 1e-8./GRAD;
        f = OLDF;
        % OPTIONS(10)=OPTIONS(10)+nvars;
        % Gradient check
        if OPTIONS(9) == 1
            GRADFD = GRAD;
            x(:)=XOUT; GRAD = eval(evalstr2);
            graderr(GRADFD, GRAD, evalstr2);
            OPTIONS(9) = 0;
        end

    else
        OPTIONS(11)=OPTIONS(11)+1;
        x(:)=XOUT; GRAD = eval(evalstr2);
    end

    %-----Initialization of Search Direction-----
    if status == -1
        SD=-GRAD;
        FIRSTF=f;
        OLDG=GRAD;
        GDOLD=GRAD'*SD;
        % For initial step-size guess assume the minimum is at zero.
        OPTIONS(18) = max(0.01, min([1,2*abs(f/GDOLD)]));
        if OPTIONS(1)>0
            % disp(sprintf('%5.0f %12.3g %12.3g ',OPTIONS(10),f,...
                % OPTIONS(18)),sprintf('%12.3g ',GDOLD));
        end
        XOUT=XOUT+OPTIONS(18)*SD;
        status=4;
        if OPTIONS(7)==0; PCNT=1; end

    else
        %-----Direction Update-----
        gdnew=GRAD'*SD;
        if OPTIONS(1)>0,
            num=[sprintf('%5.0f %12.3g %12.3g ',OPTIONS(10),f,OPTIONS(18)),...
                sprintf('%12.3g ',gdnew)];
        end
        if (gdnew>0 & f>FIRSTF)~finite(f)
            % Case 1: New function is bigger than last and gradient w.r.t. SD -ve
            % ...interpolate.
            how='inter';
            [stepsize]=cubici1(f,FIRSTF,gdnew,GDOLD,OPTIONS(18));
            if stepsize<0|isnan(stepsize), stepsize=OPTIONS(18)/2; how='C1f'; end
            if OPTIONS(18)<0.1&OPTIONS(6)==0

```

```

        if stepsize*norm(SD)<eps
            rand('normal')
            stepsize=rand(1);
            how='RANDOM STEPLENGTH!';
            status=0;
        else
            stepsize=stepsize/2;
        end
    end
    OPTIONS(18)=stepsize;
    XOUT=OLDX;
elseif f<FIRSTF
    [newstep,fbest] =cubici3(f,FIRSTF,gdnew,GDOLD,OPTIONS(18));
    sk=(XOUT-OLDX)*(GRAD-OLDG);
    if sk>1e-20
        % Case 2: New function less than old fun. and OK for updating HESS
        % .... update and calculate new direction.
        how='';
        if gdnew<0
            how='incstep';
            if newstep<OPTIONS(18)
                newstep=2*OPTIONS(18)+1e-5;
                how=[how,' IF'];
            end
            OPTIONS(18)=min([max([2,1.5*OPTIONS(18)]),1+sk+abs(gdnew)+...
                max([0,OPTIONS(18)-1]), (1.2+0.3*(~OPTIONS(7)))*abs(newstep)]);
        else % gdnew>0
            if OPTIONS(18)>0.9
                how='int st';
                OPTIONS(18)=min([1,abs(newstep)]);
            end
        end %if gdnew
        [HESS,SD]=updhess(XOUT,OLDX,GRAD,OLDG,HESS,OPTIONS);
        gdnew=GRAD'*SD;
        OLDX=XOUT;
        status=4;
    % Save Variables for next update
    FIRSTF=f;
    OLDG=GRAD;
    GDOLD=gdnew;
    % If mixed interpolation set PCNT
    if OPTIONS(7)==0, PCNT=1; MATX=zeros(3,1); MATL(1)=f; end
elseif gdnew>0 %sk<=0
    % Case 3: No good for updating HESSIAN .. interpolate or halve step length.
    how='inter st';
    if OPTIONS(18)>0.01
        OPTIONS(18)=0.9*newstep;
        XOUT=OLDX;
    end
    if OPTIONS(18)>1, OPTIONS(18)=1; end
else
    % Increase step, replace starting point
    OPTIONS(18)=max([min([newstep-OPTIONS(18),3]),0.5*OPTIONS(18)]);
    how='incst2';
    OLDX=XOUT;
    FIRSTF=f;
    OLDG=GRAD;
    GDOLD=GRAD'*SD;
    OLDX=XOUT;
end % if sk>
    % Case 4: New function bigger than old but gradient in on

```

```

% ...reduce step length.
else %gdnew<0 & F>FIRSTF
    if gdnew<0 & f>FIRSTF
        how='red step';
        if norm(GRAD-OLDG)<1e-10; HESS=eye(nvars); end
        if abs(OPTIONS(18))<eps
            rand('normal')
            SD=norm(SD)*rand(SD)
            OPTIONS(18)=abs(rand(1))*1e-6;
            how='RANDOM SD';
        else
            OPTIONS(18)=-OPTIONS(18)/2;
        end
        XOUT=OLDX;
    end %gdnew>0
end % if (gdnew>0 & F>FIRSTF)~finite(F)
XOUT=XOUT+OPTIONS(18)*SD;
if OPTIONS(1)>0
    % disp([num,how])
end
end %-----End of Direction Update-----

% Check Termination
if max(abs(SD))<2*OPTIONS(2) & (GRAD*(SD)) < 2*OPTIONS(3)
    if OPTIONS(1) > 0
        disp("");disp("");disp("");
        disp("");disp("");disp("");
        disp('Optimization Terminated Successfully');
    % disp('Gradient less than options(2)');
    disp([' NO OF ITERATIONS=', num2str(OPTIONS(10))]);
    end
    status=1;
elseif OPTIONS(10)>OPTIONS(14)
    if OPTIONS(1)>=0
        disp("");disp("");disp("");
        disp("");disp("");disp("");
        disp('Warning: Maximum number of iterations has been exceeded');
        disp(' - increase options(14) for more iterations. ');
    end
    status=1;
else
    % Line search using mixed polynomial interpolation and extrapolation.
    if PCNT~=0
        while PCNT > 0
            OPTIONS(10)=OPTIONS(10)+1;
            disp("");disp("");disp("");
            disp('Termination Check in Progress');
            disp(['Iterations: ',num2str(OPTIONS(10))]);
            x(:)=XOUT;
            f = eval(evalstr);
            [PCNT,MATL,MATX,steplen,f, how]=searchq(PCNT,f,OLDX,...
                MATL,MATX,SD,GDOLD,OPTIONS(18), how);
            OPTIONS(18)=steplen;
            XOUT=OLDX+steplen*SD;
            if abs(steplen) < 1e-6, PCNT=0; status=1; end
        end
    else
        x(:)=XOUT;
        OPTIONS(10)=OPTIONS(10)+1;
    end
end

```

```

disp('');disp('');disp('');
disp('Termination Check in Progress');
disp(['Iterations: ',num2str(OPTIONS(10))]);
    f = eval(evalstr);
    end
end
end

X(:)=XOUT;
disp('');disp('');disp('');
disp('');disp('');disp('');
disp('');disp('');disp('');
disp('Final Evaluation in Progress');
f = eval(evalstr);
if f > FIRSTF
    OPTIONS(8) = FIRSTF;
    X(:)=OLDX;
else
    OPTIONS(8) = f;
end

```

## F. MainMin

```

% Filename is "MainMin.m"
% This is the routine used by "MainOpt.m" to optimize the reference
% trajectory polynomial coefficients. It is a scaled down version
% of the dual arm spacecraft program, "Main2.m". This version does
% not integrate the state variables not include a Lyapunov controller.
% The only integration that does take place is the optimization cost
% function.

function [JOpt] = MainMin(UpCoef,ConstMat,Flags)

%clg;clear;
format compact;format short;

k = length(UpCoef);
A543 = inv(ConstMat(:,k+1:k+3))*([1; 0; 0] - ConstMat(:,1:k)*UpCoef);
Coef = [UpCoef; A543]; % Reference trajectory polynomial coefficients

% Reference Maneuver Start and Stop Times
T0 = 0;
TfR = 10;
TfC = 10;
MetaFlag = Flags(1);
ContFlag = Flags(2);
PertFlag = Flags(3);
AMatFlag = Flags(4);
WheelFlag = Flags(5);
EOMFlag = Flags(6);
PInvFlag = Flags(7);
KEFlag = Flags(8);
OutFlag = Flags(9);
Trace = Flags(10);
SymFlag = Flags(11);
ByPass = Flags(12);
TorqFlag = Flags(13);

Tol = 1e-6; % Integration tolerance

```

R2D = 180/pi; % Conversion factor from radians to degrees

% Lengths (m)

L0 = 0.75; % Origin to left shoulder  
L1 = 0.5; % Left upper arm  
L2 = 0.5; % Left forearm  
LP = 0.5; % Payload  
R2 = 0.5; % Right forearm  
R1 = 0.5; % Right upper arm  
R0 = sqrt(2\*0.75^2); % Origin to right shoulder  
Ls = [L0; L1; L2; LP; R2; R1; R0];

% Member masses (kg)

M0 = 5;  
ML1 = 1;  
ML2 = 1;  
MR1 = 1;  
MR2 = 1;  
MP = 1;  
Ms = [M0; ML1; ML2; MR1; MR2; MP];

% Center of mass distances (m)

Lc0 = 0;  
LcL1 = 0.25;  
LcL2 = 0.25;  
LcR1 = 0.25;  
LcR2 = 0.25;  
LcP = 0.25;  
CMs = [Lc0; LcL1; LcL2; LcR1; LcR2; LcP];

% MOI about center of mass:  $I_c = (1/12) * (mass) * (length)^2$

I0 = M0;  
%I0 = 0;  
IL1 = (1/12) \* ML1 \* L1^2;  
IL2 = (1/12) \* ML2 \* L2^2;  
IR1 = (1/12) \* MR1 \* R1^2;  
IR2 = (1/12) \* MR2 \* R2^2;  
IP = (1/12) \* MP \* LP^2;  
Is = [I0; IL1; IL2; IR1; IR2; IP];

% Nominal initial and desired final locations of payload

% Point Q is at wrist of left arm  
% Point P is at wrist of right arm  
Qx0n = 0.125; Qy0n = 1.5;  
Px0n = 0.625; Py0n = 1.5;  
Qxf = 0.125; Qyf = 1.0;  
Pxf = 0.125; Pyf = 1.5;

% Nominal initial and desired final locations of centerbody

Th00 = 0;  
Th0f = 0/R2D;

% Arms mount locations wrt centerbody coordinate frame (rad)

ThL0 = pi/2;  
ThR0 = pi/4;  
AngConst(1) = ThL0;  
AngConst(2) = ThR0;

% Symmetric geometry to center arms and test kinetic energy

if SymFlag  
ThL0 = 3\*pi/4;

```

AngConst(1) = ThL0;
R0 = L0; % Origin to right shoulder
l.s(7,1) = R0;
Qx0n = -0.25; Qy0n = 1.2;
Px0n = 0.25; Py0n = 1.2;
end

BoundC(1,1) = Qx0n;      BoundC(1,2) = Qy0n;
BoundC(2,1) = Px0n;      BoundC(2,2) = Py0n;
BoundC(3,1) = Qxf;       BoundC(3,2) = Qyf;
BoundC(4,1) = Pxf;       BoundC(4,2) = Pyf;
BoundC(5,1) = ThL0;      BoundC(5,2) = ThR0;
BoundC(6,1) = T0;        BoundC(6,2) = Tfr;
BoundC(7,1) = TfC;       BoundC(7,2) = ContFlag;
BoundC(8,1) = AMatFlag;   BoundC(8,2) = WheelFlag;
BoundC(9,1) = Th00;       BoundC(9,2) = Th0f;
BoundC(10,1) = EOMFlag;   BoundC(10,2) = PlnvlFlag;
BoundC(11,1) = KEFlag;    BoundC(11,2) = ByPass;
BoundC(12,1) = TorqFlag;

% Weighting Matrices
% Control torques are calculated to minimize the following cost function:
% J = 0.5*(u'*Wu*u + (A'*Lam)*Wc*(A'*Lam))
if WheelFlag
    Wu = eye(7); % Control Torque Weighting
else
    Wu = eye(6);
end
%if WheelFlag
% Wu = zeros(7,7);
%else
% Wu = zeros(6,6);
%end
%Wu(4,4)=1e5;
%Wu(7,7)=1e5;
%Wu(2,2)=1e10;
%Wu(5,5)=1e10;
Wc = zeros(8,8); % Constraint Force Weighting
%Wc = eye(8);

%%%%%%%%%%%%%%
%% INITIAL CONDITIONS %%
%%%%%%%%%%%%%%
% Desired Initial Payload Parameters
ThP0 = atan2(Py0n-Qy0n,Px0n-Qx0n);
Xc0 = 0.5 * (Px0n + Qx0n);
Yc0 = 0.5 * (Py0n + Qy0n);

Qx0 = Qx0n;
Qy0 = Qy0n;
Px0 = Px0n;
Py0 = Py0n;

% Initial State
X0 = 0;

%%%%%%%%%%%%%%
%% INTEGRATION %%
%%%%%%%%%%%%%%
% RefMin2 uses change in angular momentum to find wheel command torque
[J,Int,J] = odemin('RefMin2',T0,Tfr,X0,Tol,Trace.l.s,Ms,CMs,Is,BoundC,...

```

```

    Wu,Wc,Coef,ConstMat);
% Optimization cost function is integral of J
k = length(T);
J()pt = JInt(k);
%JOpt = max(abs(J));

```

## G. MainOpt

```

% Filename is "MainOpt.m"
% This routine optimizes the dual arm spacecraft cost function
% by changing the polynomial coefficients which describe the
% reference trajectory. It calls "Main2.m"

%clear
clc
home
format compact
format short

UpCoef0 = [0]; % Starting Guess
%UpCoef0 = UpCoef; % Use last values for starting guess
k = length(UpCoef0);
options = []; % Default values
options(1) = 0; % Display during optimization cycle: 1=On, 0=Off
options(14) = 100*k; % Maximum number of iterations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Flags during optimization %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MetaFlag = 0; % Creates metafile named "main.met"
ContFlag = 0; % Controller Status Flag: 1=On; 0=Off
PertFlag = 0; % Perturbation Flag (0=no perturbation, 1=perturbation)
                % The perturbation is to check the controller by
                % disturbing the actual initial state away from nominal.
                % The reference torques are based on nominal.
AMatFlag = 0; % Size of A matrix: 0 = 4x8 (Free Centerbody)
                % 1 = 5x8 (Fixed Centerbody)
WheelFlag = 1; % Centerbody Reaction Wheel (1=On, 0=Off)
EOMFlag = 8; % Specifies number of cost function constraint equations
                % 3 = only payload equations
                % 5 = only spacecraft equations
                % any other value = all 8 equations
PInvFlag = 1; % Psuedo-Inverse Flag (for use in finding reference torques)
                % 1 = Use psuedo-inverse
                % 0 = Use inverse
KEFlag = 0; % KE Test Flag
                % 1 = Nonzero velocity initial conditions
                % 0 = Zero velocity initial conditions
OutFlag = 0; % Output Flag
                % 1 = Display output
                % 0 = Don't display output
Trace = 1; % Observe integration
                % 1 = Observe
                % 0 = Don't observe
OptFlag = 0; % Optimization Flag
                % 1 = Perform optimization
                % 0 = Don't perform optimization
SymFlag = 0; % Symmetric Geometry Flag
                % 1 = Symmetric geometry

```

```

% 0 = Nonsymmetric geometry
ByPass = 0; % Torque calculation bypass flag
% 1 = Bypass
% 0 = Use inverse kinematics
TorqFlag = 0; % Torques to use if bypass enabled
% 0 = No Torques (Drift)
% 1 = One Shoulder Torque
% 2 = Symmetric Shoulder Torques
TorqCap = 0; % Maximum limit on wheel torque
% 1 = Enabled
% 0 = Disabled
TorqMax = 0.075; % Limit on wheel torque if TorqCap enabled
Flags1(1) = MetaFlag;
Flags1(2) = ContFlag;
Flags1(3) = PertFlag;
Flags1(4) = AMatFlag;
Flags1(5) = WheelFlag;
Flags1(6) = EOMFlag;
Flags1(7) = PlnvFlag;
Flags1(8) = KEFlag;
Flags1(9) = OutFlag;
Flags1(10) = Trace;
Flags1(11) = SymFlag;
Flags1(12) = ByPass;
Flags1(13) = TorqFlag;
Flags1(14) = TorqCap;
Flags1(15) = TorqMax;
%%%%%%%%%%%%%%
%% Flags after optimization %%
%%%%%%%%%%%%%%
Flags2 = Flags1;
Flags2(1) = 0; % MetaFlag: 1=On, 0=Off (File is "main.met")
Flags2(2) = 1; % Controller Flag: 1=On, 0=Off
Flags2(3) = 0; % Perturbation Flag: 1=On, 0=Off
Flags2(5) = 1; % Wheel Flag: 1=On, 0=Off
Flags2(8) = 0; % Kinetic Energy Flag: 1=On, 0=Off
Flags2(9) = 1; % Output Flag: 1=On, 0=Off
Flags2(10) = 1; % Trace Flag: 1=On, 0=Off
Flags2(11) = 0; % Symmetric Geometry Flag: 1=Sym, 0=NonSym
Flags2(12) = 0; % Inverse Kinematics Bypass: 1=Bypass, 0=Inverse Kinematics
Flags2(13) = 0; % Torq Flag: 0=No Torq, 1=One Torq, 2=Two Symmetric Torqs
% Torq Flag is for when the bypass is enabled
Flags2(14) = 0; % TorqCap: 1=On, 0=Off
Flags2(15) = 0.075; % Limit on maximum wheel torque
Flags2(16) = 0; % DocFlag: 1=On, 0=Off
% separate meta files for each page ("doc#.met")
DiaFlag = 0; % Diary Flag
% 1 = Create diary file "main.dia"
% 0 = No diary file

ConstMat = ones(3,k+3);
for n=1:k+3
    ConstMat(2,n) = k+6-n;
    ConstMat(3,n) = ConstMat(2,n)*(ConstMat(2,n)-1);
end
if OptFlag
    [UpCoef,options] = fminu2('MainMin',UpCoef0,options,[],ConstMat,Flags1);
end

if DiaFlag
    diary main.dia
end

```



```

if ~OptFlag
    UpCoef = UpCoef0;
end
[JInt] = Main2(UpCoef,ConstMat,Flags2);

% Plot position, velocity, & acceleration reference trajectories
k = length(UpCoef);
A543 = inv(ConstMat(:,k+1:k+3))*([1; 0; 0] - ConstMat(:,1:k)*UpCoef);
Coef = [UpCoef'; A543]; % Reference trajectory polynomial coefficients
k = length(Coef);
Steps = 21;
for m = 1:Steps
    Tau = (m-1)/(Steps-1);
    for n=1:k
        CTau(k+1-n) = Coef(k+1-n)*Tau^(n+2);
        CTaud(k+1-n) = Coef(k+1-n)*Tau^(n+1);
        CTaudd(k+1-n) = Coef(k+1-n)*Tau^n;
    end
    W(m) = ConstMat(1,:)*CTau';
    Wd(m) = ConstMat(2,:)*CTaud';
    Wdd(m) = ConstMat(3,:)*CTaudd';
end
clg
T=0:1/(Steps-1):1;
subplot(221)
plot(T,W);title('Position vs Normalized Time');
xlabel('Tau (sec)');ylabel('Position');
subplot(222)
title('Reference Trajectories')
subplot(223)
plot(T,Wd);title('Velocity vs Normalized Time');
xlabel('Tau (sec)');ylabel('Velocity');
subplot(224)
plot(T,Wdd);title('Acceleration vs Normalized Time');
xlabel('Tau (sec)');ylabel('Acceleration');
if Flags2(1)
    meta main
end
if Flags2(16)
    meta doc6
end
pause

disp('Initial guess for highest order coefficients');disp(UpCoef0);
disp('Coefficients in descending order');disp(Coef);
disp('Integral of Cost Function,(JIntAbs & JIntSqr)');disp(JInt);
if OptFlag
    disp('Iterations');disp(options(10));
end
diary off

```

## H. Main2

```

% Filename is "Main2.m"
% This routine is the driver for the dual arm spacecraft problem
% but is called by "MainOpt.m" after the polynomial reference trajectory
% coefficients have been optimized.

```

```

function [JIntTotal] = Main2(UpCoef,ConstMat,Flags)

% Calculate the coefficients for orders five, four, and three.
% Include these with the higher order coefficients in a vector.
k = length(UpCoef);
A543 = inv(ConstMat(:,k+1:k+3))*([1; 0; 0] - ConstMat(:,1:k)*UpCoef');
Coef = [UpCoef'; A543]; % Reference trajectory polynomial coefficients

%%%%%%%%%%%%
%% Times %%
%%%%%%%%%%%%
% Reference Maneuver Start and Stop Times and Controller Stop Times
% Setting the controller time longer than the reference maneuver time
% ensures that the controller eliminates any errors remaining after the
% reference trajectory should be complete. To exercise the controller
% only with no reference trajectory, set the reference maneuver stop
% time to a negative value.
T0 = 0;
TfR = 10;
TfC = 10;
MetaFlag = Flags(1);
ContFlag = Flags(2);
PertFlag = Flags(3);
AMatFlag = Flags(4);
WheelFlag = Flags(5);
EOMFlag = Flags(6);
PInvFlag = Flags(7);
KEFlag = Flags(8);
OutFlag = Flags(9);
Trace = Flags(10);
SymFlag = Flags(11);
ByPass = Flags(12);
TorqFlag = Flags(13);
TorqCap = Flags(14);
TorqMax = Flags(15);
DocFlag = Flags(16);

Pert = -10; % Perturbation of initial payload angle, ThetaP (deg)
Tol = 1e-6; % Integration tolerance
Interval = 3; % Stick figure drawing includes every Interval'th time
R2D = 180/pi; % Conversion factor from radians to degrees

%%%%%%%%%%%%
%% System Parameters %%
%%%%%%%%%%%%
% Lengths (m)
L0 = 0.75; % Origin to left shoulder
L1 = 0.5; % Left upper arm
L2 = 0.5; % Left forearm
LP = 0.5; % Payload
R2 = 0.5; % Right forearm
R1 = 0.5; % Right upper arm
R0 = sqrt(2*0.75^2); % Origin to right shoulder
Ls = [L0; L1; L2; LP; R2; R1; R0];

% Member masses (kg)
M0 = 5;
ML1 = 1;
ML2 = 1;
MR1 = 1;

```

```

MR2 = 1;
MP = 1;
Ms = [M0; ML1; ML2; MR1; MR2; MP];

% Center of mass distances (m)
Lc0 = 0;
LcL1 = 0.25;
LcL2 = 0.25;
LcR1 = 0.25;
LcR2 = 0.25;
LcP = 0.25;
CMs = [Lc0; LcL1; LcL2; LcR1; LcR2; LcP];

% MOI about individual centers of mass
% Arms are modelled as slender rods: Ic = (1/12)*(mass)*(length)^2
I0 = M0;
IL1 = (1/12) * ML1 * L1^2;
IL2 = (1/12) * ML2 * L2^2;
IR1 = (1/12) * MR1 * R1^2;
IR2 = (1/12) * MR2 * R2^2;
IP = (1/12) * MP * LP^2;
Is = [I0; IL1; IL2; IR1; IR2; IP];

% Nominal initial and desired final locations of payload
% Point Q is at wrist of left arm
% Point P is at wrist of right arm
Qx0n = 0.125; Qy0n = 1.5;
Px0n = 0.625; Py0n = 1.5;
Qxf = 0.125; Qyf = 1.0;
Pxf = 0.125; Pyf = 1.5;

% Nominal initial and desired final locations of centerbody
Th00 = 0/R2D;
Th0f = 0/R2D;

% Arms mount locations wrt centerbody coordinate frame (rad)
ThL0 = pi/2;
ThR0 = pi/4;
AngConst(1) = ThL0;
AngConst(2) = ThR0;

% Symmetric geometry to center arms and test kinetic energy
if SymFlag
ThL0 = 3*pi/4;
AngConst(1) = ThL0;
R0 = 1.0; % Origin to right shoulder
Is(7,1) = R0;
Qx0n = -0.25; Qy0n = 1.2;
Px0n = 0.25; Py0n = 1.2;
end

% Assemble information required in other subroutines into a matrix
BoundC(1,1) = Qx0n; BoundC(1,2) = Qy0n;
BoundC(2,1) = Px0n; BoundC(2,2) = Py0n;
BoundC(3,1) = Qxf; BoundC(3,2) = Qyf;
BoundC(4,1) = Pxf; BoundC(4,2) = Pyf;
BoundC(5,1) = ThL0; BoundC(5,2) = ThR0;
BoundC(6,1) = T0; BoundC(6,2) = TfR;
BoundC(7,1) = TfC; BoundC(7,2) = ContFlag;
BoundC(8,1) = AMatFlag; BoundC(8,2) = WheelFlag;
BoundC(9,1) = Th00; BoundC(9,2) = Th0f;

```

```

BoundC(10,1)= EOMFlag; BoundC(10,2)= PInvFlag;
BoundC(11,1)= KEFlag; BoundC(11,2)= Bypass;
BoundC(12,1)= TorqFlag;
BoundC(13,1)= TorqCap; BoundC(13,2)= TorqMax;

% Gp are gains for angle i position error
% Giv are gains for angle i velocity error
Gpos = 0.5; % Position error gain
Gvel = 0.2; % Velocity error gain
Gains = [Gpos; Gvel];

% Weighting Matrices
% Control torques are calculated to minimize the following cost function:
%  $J = 0.5(u^*Wu*u + (A^*Lam)^*Wc*(A^*Lam))$ 
% Wu is the control torque weighting matrix
% Wc is the constraint force weighting matrix
if WheelFlag
    Wu = eye(7); % Control Torque Weighting
else
    Wu = eye(6);
end
%if WheelFlag
% Wu = zeros(7,7);
%else
% Wu = zeros(6,6);
%end
%Wu(4,4)=1e5; % Penalty on wrist motors for free centerbody case
%Wu(7,7)=1e5;
%Wu(2,2)=1e10; % Penalty on wrist motors for fixed centerbody case
%Wu(5,5)=1e10;
Wc = zeros(8,8); % Constraint Force Weighting
%Wc = eye(8);

%%%%%%%%%%%%%%
%% INITIAL CONDITIONS %%
%%%%%%%%%%%%%%
% Desired Initial Payload Parameters
ThP0 = atan2(Py0n-Qy0n,Px0n-Qx0n);
Xc0 = 0.5 * (Px0n + Qx0n);
Yc0 = 0.5 * (Py0n + Qy0n);

if PertFlag % Perturbation enabled
    ThP0 = ThP0 + Pert/R2D; % Perturb payload angle
    Qx0 = Xc0 - LcP*cos(ThP0); % Perturb arm end points
    Qy0 = Yc0 - LcP*sin(ThP0);
    Px0 = Xc0 + (LP-LcP)*cos(ThP0);
    Py0 = Yc0 + (LP-LcP)*sin(ThP0);
else % No Perturbation
    Qx0 = Qx0n;
    Qy0 = Qy0n;
    Px0 = Px0n;
    Py0 = Py0n;
end
PertCrd = [Qx0 Qy0 Px0 Py0];

% Left Arm
% Elbow is left of line from arm base to Q (RQ)
LSx = L0 * cos(Th00 + ThL0);
LSy = L0 * sin(Th00 + ThL0);
RQ = sqrt((Qx0-LSx)^2+(Qy0-LSy)^2); % Length from arm base to Q
Beta1 = atan2(Qy0-LSy,Qx0-LSx); % Angle from arm base to RQ

```

```

% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RQ and Link L1
Num = L1^2 + RQ^2 - L2^2;
Den = 2 * L1 * RQ;
Beta2 = acos(Num/Den); % Angle from RQ to Link 1
ThL10 = (Beta1 + Beta2) - (Th00 + ThL0); % Theta L1
% Use law of cosines to find the interior angle at the elbow
Num = L1^2 + L2^2 - RQ^2;
Den = 2 * L1 * L2;
Beta3 = acos(Num/Den);
ThL20 = -(pi-Beta3);

% Right Arm
% Elbow is right of line from arm base (shoulder) to P (wrist) (RP)
RSx = R0 * cos(Th00 + ThR0);
RSy = R0 * sin(Th00 + ThR0);
RP = sqrt((Px0-RSx)^2+(Py0-RSy)^2); % Length from arm base to P
Beta1 = atan2(Py0-RSy,Px0-RSx); % Angle from arm base to RP
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RP and Link R1
Num = R1^2 + RP^2 - R2^2;
Den = 2 * R1 * RP;
Beta2 = acos(Num/Den); % Angle from Link R1 to RP
Beta4 = Beta1 - (Th00 + ThR0);
ThR10 = -(Beta2 - Beta4);
Num = R1^2 + R2^2 - RP^2;
Den = 2 * R1 * R2;
Beta3 = acos(Num/Den);
ThR20 = pi - Beta3;

% Desired Initial State
X0 = [Th00; ThL10; ThL20; ThR10; ThR20; ThP0; Xc0; Yc0;...
      0; 0; 0; 0; 0; 0; 0; 0];

%%%%%%%%%%%%%%
%% Kinetic Energy Test Conditions %%
%%%%%%%%%%%%%%
% Specify Payload and Centerbody Initial Rates
% Compatible Rates for the Redundant Coordinates are calculated
if KEFlag
    ThPd0 = 0/R2D; % Rates to specify
    Xcd0 = -0.03;
    Ycd0 = -0.03;
    Th0d0 = 0/R2D;
    %%%%%%%%%%%%%%%
    %% LEFT ARM %%
    %%%%%%%%%%%%%%%
    % [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd
    % Qxd & Qyd are x & y components of point Q inertial velocity.
    % Thd = [ThL1dot; ThL2dot]
    % H matrices are made from expressing the x & y components of Q in
    % terms of L0, Th0, ThL0, L1, ThL1, L2, and ThL2.
    % Qx=L0*cos(Th0+ThL0)+L1*cos(Th0+ThL0+ThL1)+L2*cos(Th0+...
    % ThL0+ThL1+ThL2)
    % Qy=L0*sin(Th0+ThL0)+L1*sin(Th0+ThL0+ThL1)+L2*sin(Th0+...
    % ThL0+ThL1+ThL2)
    % The differentiation of these equations lead to
    % [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd which can be solved for Thd
    Qxd0 = Xcd0 + LcP * ThPd0 * sin(ThP0);
    Qyd0 = Ycd0 - LcP * ThPd0 * cos(ThP0);
    H2(1,2) = -L2*sin(Th00+ThL0+ThL10+ThL20);

```

```

H2(1,1) = H2(1,2) - L1*sin(Th00+ThL0+ThL10);
H2(2,2) = L2*cos(Th00+ThL0+ThL10+ThL20);
H2(2,1) = H2(2,2) + L1*cos(Th00+ThL0+ThL10);
H1(1,1) = H2(1,1) - L0*sin(Th00+ThL0);
H1(2,1) = H2(2,1) + L0*cos(Th00+ThL0);
Thd0 = inv(H2) * ([Qxd0; Qyd0] - H1*Th0d0);
% Angle rates
ThL1d0 = Thd0(1);
ThL2d0 = Thd0(2);
%%%%%%%%%%
%% RIGHT ARM %%
%%%%%%%%%%
% The development is similar to the left arm
% Px=R0*cos(Th0+ThR0)+R1*cos(Th0+ThR0+ThR1)+R2*cos(Th0+...
% ThR0+ThR1+ThR2)
% Py=R0*sin(Th0+ThR0)+R1*sin(Th0+ThR0+ThR1)+R2*sin(Th0+...
% ThR0+ThR1+ThR2)
% [Pxd; Pyd] = [H1]*Th0d + [H2]*Thd
Pxd0 = Xcd0 - (LP - LcP) * ThPd0 * sin(ThP0);
Pyd0 = Ycd0 + (LP - LcP) * ThPd0 * cos(ThP0);
H2(1,2) = -R2*sin(Th00+ThR0+ThR10+ThR20);
H2(1,1) = H2(1,2) - R1*sin(Th00+ThR0+ThR10);
H2(2,2) = R2*cos(Th00+ThR0+ThR10+ThR20);
H2(2,1) = H2(2,2) + R1*cos(Th00+ThR0+ThR10);
H1(1,1) = H2(1,1) - R0*sin(Th00+ThR0);
H1(2,1) = H2(2,1) + R0*cos(Th00+ThR0);
Thd0 = inv(H2) * ([Pxd0; Pyd0] - H1*Th0d0);
% Angle rates
ThR1d0 = Thd0(1);
ThR2d0 = Thd0(2);
X0 = {Th00; ThL10; ThL20; ThR10; ThR20; ThP0; Xc0; Yc0;...
Th0d0; ThL1d0; ThL2d0; ThR1d0; ThR2d0; ThPd0; Xcd0; Ycd0};
end

%%%%%%%%%%
%% FINAL CONDITIONS %%
%%%%%%%%%%
% Desired Final Payload Angle
ThPf = atan2(Pyf-Qyf,Pxf-Qxf);

% Left Arm
% Elbow is left of line from arm base to Q (RQ)
LSx = L0 * cos(Th0f + ThL0);
LSy = L0 * sin(Th0f + ThL0);
RQ = sqrt((Qxf-LSx)^2+(Qyf-LSy)^2); % Length from arm base to Q
Beta1 = atan2(Qyf-LSy,Qxf-LSx); % Angle from arm base to RQ
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RQ and Link L1
Num = L1^2 + RQ^2 - L2^2;
Den = 2 * L1 * RQ;
Beta2 = acos(Num/Den); % Angle from RQ to Link 1
ThL1f = (Beta1 + Beta2) - (Th0f + ThL0); % Theta L1
% Use law of cosines to find the interior angle at the elbow
Num = L1^2 + L2^2 - RQ^2;
Den = 2 * L1 * L2;
Beta3 = acos(Num/Den);
ThL2f = -(pi-Beta3);

% Right Arm
% Elbow is right of line from arm base (shoulder) to P (wrist) (RP)
RSx = R0 * cos(Th0f + ThR0);

```

```

RSy = R0 * sin(Th0f + ThR0);
RP = sqrt((Pxf-RSx)^2+(Pyf-RSy)^2); % Length from arm base to P
Beta1 = atan2(Pyf-RSy,Pxf-RSx); % Angle from arm base to RP
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RP and Link R1
Num = R1^2 + RP^2 - R2^2;
Den = 2 * R1 * RP;
Beta2 = acos(Num/Den); % Angle from Link R1 to RP
Beta4 = Beta1 - (Th0f + ThR0);
ThR1f = -(Beta2 - Beta4);
Num = R1^2 + R2^2 - RP^2;
Den = 2 * R1 * R2;
Beta3 = acos(Num/Den);
ThR2f = pi - Beta3;

% Desired Final State
Xcf = 0.5 * (Pxf + Qxf);
Ycf = 0.5 * (Pyf + Qyf);
QfDes = [Th0f; ThL1f; ThL2f; ThR1f; ThR2f; ThPf; Xcf; Ycf;...
0; 0; 0; 0; 0; 0; 0; 0];

if OutFlag
%%%%%%%%%%%%%%
%% PROBLEM SUMMARY %%
%%%%%%%%%%%%%%
disp('PROBLEM SUMMARY')
disp('')
disp('Initial Angles (deg)')
disp('Initial Angular Rates (deg/sec)')
disp('Desired Final Angles (deg)')
disp(' Theta0 ThetaL1 ThetaL2 ThetaR1 ThetaR2 ThetaP')
disp(X0(1:6)*R2D)
disp(X0(9:14)*R2D)
disp(QfDes(1:6)*R2D)
disp('')
disp('Payload Coordinates (m)')
disp(' Nominal Initial, Perturbed Initial, and Final')
disp(' Qx Qy Px Py')
TableCrd = [Qx0n Qy0n Px0n Py0n; PertCrd; Qxf Qyf Pxf Pyf];
disp(TableCrd)
disp('')
disp('Arm Mounting Locations wrt Centerbody Coordinate Frame (deg)')
disp(ThL0*R2D);disp(ThR0*R2D)
disp('')
disp('Start, Reference Manuever Stop, & Simulation Stop Times (sec)')
disp(T0);disp(TfR);disp(TfC)
disp('')
disp('Controller Status (1 = On; 0 = Off)')
disp(ContFlag)
disp('')
disp('Perturbation Status (1 = On; 0 = Off)')
disp(PertFlag)
disp('')
disp('Centerbody Status in Forward EOM (1 = Fixed; 0 = Free)')
disp(AMatFlag)
disp('')
disp('Reaction Wheel Status (1 = On; 0 = Off)')
disp(WheelFlag)
disp('')
disp('Number of Equations in Cost Function Constraint (3, 5 or 8)')
disp(EOMFlag)

```

```

disp("")
disp('Pseudo-Inverse Status (1 = On; 0 = Off)')
disp(PInvFlag)
disp("")
disp('Nonzero Initial Velocity Status (1 = On; 0 = Off)')
disp(KEFlag)
disp("")
disp('Geometry Status (1 = Symmetric; 0 = Nonsymmetric)')
disp(SymFlag)
disp("")
disp('Inverse Kinematics Bypass Status (1 = Bypass; 0 = Use inv. kinematics)')
disp(ByPass)
disp("")
disp('Torques if Bypass Enabled (0=None, 1=One, 2=Two Symmetric)')
disp(TorqFlag)
disp("")
disp('Reaction Wheel Torque Cap Status (1=Enabled, 0=Disabled)')
disp(TorqCap)
if TorqCap
    disp('Limit on Wheel Torque');
    disp(TorqMax);
end
disp("")
disp('Controller Gains (position and velocity)')
disp('  Gpos  Gvel')
disp(Gains)
disp("")
disp('Cost Function: J = 0.5*(uT*Wu*u + (AT*Lam)T*Wc*(AT*Lam))')
disp('  where __T signifies transpose')
disp('Control Torques Weighting Matrix, Wu')
disp(Wu)
%disp('Constraint Forces Weighting Matrix, Wc')
%disp(Wc)

end % End of OutFlag branch

%%%%%%%%%%%%%%
%% INTEGRATION %%
%%%%%%%%%%%%%%
% "ode" is a variable step size Runge-Kutta integrator function
% supplied with MATLAB. "ode2" is the same as "ode" in its function
% but permits the passing of more variables into and out of the function.
[T,X,Torq,TorqRef,Aqdot,J,Res,LHS,RHS,Delq] = ...
    ode2('Eqn2',T0,TfC,X0,Tol,Trace,Ls,Ms,CMs,Is,BoundC,...
        Gains,QfDes,Wu,Wc,Coef,ConstMat);
k = length(T);
JInt = X(:,17:18);
JIntTotal = X(k,17:18);

if OutFlag
%%%%%%%%%%%%%%
%% OUTPUT %%
%%%%%%%%%%%%%%

clg;
% Angle Histories
n = length(T);
Q = X(:,1:6);
subplot(221)
plot(T,Q(:,1)*R2D,T,Q(:,2)*R2D,T,Q(:,3)*R2D,...
    T,Q(:,4)*R2D,T,Q(:,5)*R2D,T,Q(:,6)*R2D);

```



```

hold on
plot(T(n),QfDes(1)*R2D,'*',T(n),QfDes(2)*R2D,'*',T(n),QfDes(3)*R2D,'*',...
     T(n),QfDes(4)*R2D,'*',T(n),QfDes(5)*R2D,'*',T(n),QfDes(6)*R2D,'*');
title('Thetas vs Time');
xlabel('Time (sec)');ylabel('Angles (deg)');
hold off
% Angle Rate Histories
Qdot = X(:,9:14);
subplot(223)
plot(T,Qdot(:,1)*R2D,T,Qdot(:,2)*R2D,T,Qdot(:,3)*R2D,...
     T,Qdot(:,4)*R2D,T,Qdot(:,5)*R2D,T,Qdot(:,6)*R2D);
title('ThetaDots vs Time');
xlabel('Time (sec)');ylabel('Angle Rates (deg/sec)');

%Departures from Reference Trajectory
if ~ByPass
    subplot(222)
    plot(T,Delq(1,:)*R2D,T,Delq(2,:)*R2D,T,Delq(3,:)*R2D,...
         T,Delq(4,:)*R2D,T,Delq(5,:)*R2D,T,Delq(6,:)*R2D);
    title('Displacement Errors vs Time');
    xlabel('Time (sec)');ylabel('Q-QRef (deg)');
end

if MetaFlag
    meta main
end
if DocFlag
    meta doc1
end
pause

% Draw Motion
Angles = Q(:,1:6);
[Xcoord,Ycoord] = Draw3(Ls,AngConst,Angles,Interval);
if MetaFlag
    meta main
end
if DocFlag
    meta doc2
end
pause

disp("");
disp('STATE VECTOR TIME HISTORY:');
disp('Angles (deg)')
Table1 = [T X(:,1:6)*R2D];
disp(' Time  Theta0  ThetaL1  ThetaL2  ThetaR1  ThetaR2  ThetaP');
disp(Table1)
pause

disp("");
disp('Angle Rates (deg/sec)')
Table2 = [T Qdot(:,1:6)*R2D];
disp(' Time  Th0dot  ThL1dot  ThL2dot  ThR1dot  ThR2dot  ThPdot');
disp(Table2)
pause

if ~ByPass
    disp("");
    disp('TRAJECTORY ERROR TIME HISTORY:');
    disp('Angles (deg)')

```

```

    Table2a = [T R2D*Delq(1:6,:)'];
    disp('   Time   DelTh0 DelThL1 DelThL2 DelThR1 DelThR2 DelThP');
    disp(Table2a)
end
pause

% Reference Torque Histories
clg;
if TIR > 0
    if TIR < TFC
        [r,s] = size(TorqRef);
        TorqRef = [TorqRef zeros(s,1)];
        TRef = [T(1:r); TIR];
    else
        TRef = T;
    end
    subplot(221)
    plot(TRef,TorqRef);
    title('Reference Torques vs Time');
    xlabel('Time (sec)');ylabel('Reference Torques');
end
% Command Torque Histories
%Torq = [Torq, zeros(4,1)];
k=n;
subplot(223)
plot(T(1:k),Torq);
title('Command Torques vs Time');
xlabel('Time (sec)');ylabel('Command Torques');

% Cost Function
subplot(222)
plot(T,J(1,:));title('Cost vs Time');
xlabel('Time (sec)');ylabel('J=abs(Uwh)');
subplot(224)
plot(T,JInt);title('Integrated Cost vs Time');
xlabel('Time (sec)');ylabel('JInt');
if MetaFlag
    meta main
end
if DocFlag
    meta doc3
end
pause

if TIR > 0
    disp('')
    disp('REFERENCE TORQUE HISTORY');
    if WheelFlag
        disp('   Time   U0  ULS  ULE  ULW  URS  URE  URW');
    else
        disp('   Time   ULS  ULE  ULW  URS  URE  URW');
    end
    Table4 = [TRef TorqRef];
    disp(Table4)
end
pause
disp('')
disp('COMMAND TORQUE HISTORY');
if WheelFlag
    disp('   Time   U0  ULS  ULE  ULW  URS  URE  URW');
else

```

```

disp(' Time ULS ULE ULW URS URE URW');
end
Table5 = [T(1:k) Torq'];
disp(Table5)
pause

Table6 = [T(1:k) J(1,:) JInt];
disp('');
disp('COST FUNCTION HISTORY');
disp(' Time J JIntAbs JIntSqr');
disp(Table6);
pause

% Angular Momentum
k = length(T);
for n = 1:k
    [Hs] = AngMo(Ls,Ms,CMs,Is,X(n,1:8),X(n,9:16));
    if n == 1
        HHist = Hs;
    else
        HHist = [HHist; Hs];
    end
end
clf
subplot(221);
plot(T,HHist(:,1:6));title('Angular Momentum of Pieces vs Time');
xlabel('Time (sec)');ylabel('Ang Momentum (N-m-sec)');
subplot(223);
plot(T,HHist(:,7));title('Total Angular Momentum vs Time');
xlabel('Time (sec)');ylabel('Ang Momentum (N-m-sec)');
% Kinetic Energy
for m = 1:k
    if AMatFlag
        [M,G,A,Adot,B] = MatxFix(Ls,Ms,CMs,Is,X(m,1:8),X(m,9:16),AngConst);
    else
        [M,G,A,Adot,B] = Matx(Ls,Ms,CMs,Is,X(m,1:8),X(m,9:16),AngConst);
    end
    LHSTot(m) = 0;
    RHSTot(m) = 0;
    ResTot(m) = 0;
    for n=1:8
        LHSTot(m) = LHSTot(m) + LHS(n,m);
        RHSTot(m) = RHSTot(m) + RHS(n,m);
    end
    ResTot(m) = LHSTot(m) - RHSTot(m);
    KE(m) = 0.5*X(m,9:16)*M*X(m,9:16)';
end
subplot(224)
plot(T,KE);title('Kinetic Energy vs Time');
xlabel('Time (sec)');ylabel('KE (kg m^2/s^2)');
% Compare wheel torque to change in total angular momentum
Hd = J(2,:);
subplot(222)
plot(T(1:k)',Torq(1,:),T(1:k)',Hd);
title('Compare Wheel Torque to Change in Ang. Mom. ');
xlabel('Time (sec)');
pause
if MetaFlag
    meta main
end
if DocFlag

```

```

    meta doc4
end
%pause

clf
subplot(221)
plot(T,Res);title('Residuals of Equations');
xlabel('Time (sec)');ylabel('LHS-RHS');
subplot(223)
plot(T,ResTot);title('Total Residuals');
xlabel('Time (sec)');ylabel('LHS-RHS');
% Constraints: see if  $A \cdot \dot{Q} = 0$  is satisfied
subplot(222)
plot(T(1:k),Aqdot(1,:),T(1:k),Aqdot(2,:),T(1:k),Aqdot(3,:),...
    T(1:k),Aqdot(4,:));
[dum1,dum2] = size(Aqdot);
if dum1 ==5
    hold on
    plot(T(1:k),Aqdot(5,:));
    hold off
end
title('Constraints:  $A \cdot \dot{Q}$  vs Time');
xlabel('Time (sec)');ylabel('A * Qdot');
if MetaFlag
    meta main
end
if DocFlag
    meta doc5
end
pause

end % End of OutFlag branch

```

## I. Matx

```

% Filename is 'Matx.m'
% This routine calculates the matrices for the dual arm
% spacecraft EOM when it is grasping a payload. Each arm
% has two links. This version assumes that the centerbody
% is NOT fixed. This impacts A and Adot.

function [M,G,A,Adot,B] = Matx(Ls,Ms,CMs,Is,Ths,Thdots,AngConst)

% OUTPUTS:
% M = 8x8 mass matrix
% G = 8x1 vector with coriolis and centripetal terms
% A = 4x8 constraints matrix
% Adot = 4x8 derivative of constraints matrix
% B = Control influence matrix
%
% INPUTS:
% Ls = 7x1 vector of lengths (m)
% 1st element = distance from origin to left arm mount
% 2nd & 3rd elements wrt left arm (from shoulder to wrist)
% 4th element = payload length
% 5th & 6th elements wrt right arm (from wrist to shoulder)
% 7th element = distance from right arm mount to origin
% [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)

```

```

%      1st element = mass of spacecraft centerbody
%      2nd & 3rd elements = left arm (upper then lower arm)
%      4th & 5th elements = right arm (upper then lower arm)
%      6th element = payload mass
%      [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations (m)
%      [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% Is = 6x1 column vector containing the moments of inertias
%      about the respective body's center of mass (kg m^2)
%      1st element = inertia of spacecraft centerbody
%      2nd & 3rd elements = left arm (upper then lower arm)
%      4th & 5th elements = right arm (upper then lower arm)
%      6th element = payload inertia
%      [I0; IL1; IL2; IR1; IR2; IP]
% Ths = 6 element vector containing the angles which describe
%      the configuration of the system.
%      [Th0; ThL1; ThL2; ThR1; ThR2; ThP]
% Thdots = 6 element vector containing the angle rates
% AngConst = 2 element vector of arm mounting locations
%      [ThL0; ThR0]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lengths (m)
L0 = Is(1);
L1 = Is(2);
L2 = Is(3);
LP = Is(4);
R2 = Is(5);
R1 = Is(6);
R0 = Is(7);

% Member masses (kg)
M0 = Ms(1);
ML1 = Ms(2);
ML2 = Ms(3);
MR1 = Ms(4);
MR2 = Ms(5);
MP = Ms(6);

% Center of mass distances (m)
Lc0 = CMs(1);
LcL1 = CMs(2);
LcL2 = CMs(3);
LcR1 = CMs(4);
LcR2 = CMs(5);
LcP = CMs(6); %measured from left end

% MOI about center of mass
I0 = Is(1);
IL1 = Is(2);
IL2 = Is(3);
IR1 = Is(4);
IR2 = Is(5);
IP = Is(6);

% Angles
Th0 = Ths(1);
ThL1 = Ths(2);
ThL2 = Ths(3);

```

```

ThR1 = Ths(4);
ThR2 = Ths(5);
ThP = Ths(6);

% Angle Rates
Th0d = Thdots(1);
ThL1d = Thdots(2);
ThL2d = Thdots(3);
ThR1d = Thdots(4);
ThR2d = Thdots(5);
ThPd = Thdots(6);

% Arm mount locations
ThL0 = AngConst(1);
ThR0 = AngConst(2);

%%%%%%%%%%%%%%
%% Mass Matrix %%
%%%%%%%%%%%%%%
M = zeros(8,8);
M(8,8) = MP;
M(7,7) = MP;
M(6,6) = IP;
M(5,5) = IR2 + MR2*LcR2^2;
M(5,4) = M(5,5) + MR2*R1*LcR2*cos(ThR2);
M(4,5) = M(5,4);
M(5,1) = M(4,5) + MR2*R0*LcR2*cos(ThR1+ThR2);
M(1,5) = M(5,1);
M(4,4) = M(4,5)+IR1+MR2*R1*LcR2*cos(ThR2)+MR1*LcR1^2+MR2*R1^2;
M(4,1)=M(4,4)+R0*(MR1*LcR1+MR2*R1)*cos(ThR1)+MR2*R0*LcR2*...
    cos(ThR1+ThR2);
M(1,4) = M(4,1);
M(3,3) = IL2 + ML2*LcL2^2;
M(3,2) = M(3,3) + ML2*L1*LcL2*cos(ThL2);
M(2,3) = M(3,2);
M(3,1) = M(3,2) + ML2*L0*LcL2*cos(ThL1+ThL2);
M(1,3) = M(3,1);
M(2,2) = M(3,2)+ML2*L1*LcL2*cos(ThL2)+IL1+ML1*LcL1^2+ML2*L1^2;
M(2,1)=M(2,2)+L0*(ML1*LcL1+ML2*L1)*cos(ThL1)+ML2*L0*LcL2*...
    cos(ThL1+ThL2);
M(1,2) = M(2,1);
Part1 = I0+M(2,2)+M(4,4)+M0*Lc0^2+(ML1+ML2)*I0^2+(MR1+MR2)*R0^2;
Part2 = 2*L0*(ML1*LcL1+ML2*L1)*cos(ThL1)+2*ML2*L0*LcL2*...
    cos(ThL1+ThL2);
Part3 = 2*R0*(MR1*LcR1+MR2*R1)*cos(ThR1)+2*MR2*R0*LcR2*...
    cos(ThR1+ThR2);
M(1,1) = Part1 + Part2 + Part3;

%%%%%%%%%%%%%%
%% G %%
%%%%%%%%%%%%%%
G = zeros(8,1);
Pt1 = -L0*(ThL1d^2+2*Th0d*ThL1d)*(ML1*LcL1+ML2*L1)*sin(ThL1);
Pt2 = -ML2*L1*LcL2*ThL2d*(2*Th0d+2*ThL1d+ThL2d)*sin(ThL2);
Pt3=-ML2*L0*LcL2*(2*Th0d*(ThL1d+ThL2d)+(ThL1d+ThL2d)^2)*...
    sin(ThL1+ThL2);
Pt4 = -R0*(ThR1d^2+2*Th0d*ThR1d)*(MR1*LcR1+MR2*R1)*sin(ThR1);
Pt5 = -MR2*R1*LcR2*ThR2d*(2*Th0d+2*ThR1d+ThR2d)*sin(ThR2);
Pt6=-MR2*R0*LcR2*(2*Th0d*(ThR1d+ThR2d)+(ThR1d+ThR2d)^2)*...
    sin(ThR1+ThR2);
G(1) = Pt1 + Pt2 + Pt3 + Pt4 + Pt5 + Pt6;

```

```

Pt1 = L0*Th0d^2*(ML1*LcL1+ML2*L1)*sin(ThL1);
Pt2 = -ML2*L1*LcL2*ThL2d*(2*Th0d+2*ThL1d+ThL2d)*sin(ThL2);
Pt3 = ML2*L0*LcL2*Th0d^2*sin(ThL1+ThL2);
Gi(2) = Pt1 + Pt2 + Pt3;
Gi(3) = ML2*LcL2*(L1*(Th0d+ThL1d)^2*sin(ThL2)+L0*Th0d^2*...
    sin(ThL1+ThL2));
Pt1 = R0*Th0d^2*(MR1*LcR1+MR2*R1)*sin(ThR1);
Pt2 = -MR2*R1*LcR2*ThR2d*(2*Th0d+2*ThR1d+ThR2d)*sin(ThR2);
Pt3 = MR2*R0*LcR2*Th0d^2*sin(ThR1+ThR2);
Gi(4) = Pt1 + Pt2 + Pt3;
Gi(5) = MR2*LcR2*(R1*(Th0d+ThR1d)^2*sin(ThR2)+R0*Th0d^2*...
    sin(ThR1+ThR2));

```

```

%%%%%%%%%%
%% Constraints Matrix %%
%%%%%%%%%%

```

```

% The constraint matrix comes from putting the constraint equations
% into the Pfaffian form: A*qdot + A0 = 0. The first two constraint
% equations are found by finding the x and y components of the
% Payload's center of mass by starting at the origin and moving
% up the left arm. The second two constraint equations find the x
% and y components of the Payload's center of mass by starting at the
% origin and moving to the base of the right arm and then
% up the right arm. Differentiating these equations results
% in the Pfaffian form with A0 = 0.

```

```

A = zeros(4,8);
A(1,7) = -1;
A(2,8) = -1;
A(3,7) = -1;
A(4,8) = -1;
A(1,6) = -LcP*sin(ThP);
A(2,6) = LcP*cos(ThP);
A(3,6) = (LP-LcP)*sin(ThP);
A(4,6) = -(LP-LcP)*cos(ThP);
A(4,5) = R2*cos(Th0+ThR0+ThR1+ThR2);
A(4,4) = A(4,5) + R1*cos(Th0+ThR0+ThR1);
A(4,1) = A(4,4) + R0*cos(Th0+ThR0);
A(3,5) = -R2*sin(Th0+ThR0+ThR1+ThR2);
A(3,4) = A(3,5) - R1*sin(Th0+ThR0+ThR1);
A(3,1) = A(3,4) - R0*sin(Th0+ThR0);
A(2,3) = L2*cos(Th0+ThL0+ThL1+ThL2);
A(2,2) = A(2,3) + L1*cos(Th0+ThL0+ThL1);
A(2,1) = A(2,2) + L0*cos(Th0+ThL0);
A(1,3) = -L2*sin(Th0+ThL0+ThL1+ThL2);
A(1,2) = A(1,3) - L1*sin(Th0+ThL0+ThL1);
A(1,1) = A(1,2) - L0*sin(Th0+ThL0);

```

```

Adot = zeros(4,8);
Adot(1,6) = -ThPd*LcP*cos(ThP);
Adot(2,6) = -ThPd*LcP*sin(ThP);
Adot(3,6) = ThPd*(LP-LcP)*cos(ThP);
Adot(4,6) = ThPd*(LP-LcP)*sin(ThP);
Adot(4,5) = -(Th0d+ThR1d+ThR2d)*R2*sin(Th0+ThR0+ThR1+ThR2);
Adot(4,4) = Adot(4,5) - (Th0d+ThR1d)*R1*sin(Th0+ThR0+ThR1);
Adot(4,1) = Adot(4,4) - Th0d*R0*sin(Th0+ThR0);
Adot(3,5) = -(Th0d+ThR1d+ThR2d)*R2*cos(Th0+ThR0+ThR1+ThR2);
Adot(3,4) = Adot(3,5) - (Th0d+ThR1d)*R1*cos(Th0+ThR0+ThR1);
Adot(3,1) = Adot(3,4) - Th0d*R0*cos(Th0+ThR0);
Adot(2,3) = -(Th0d+ThL1d+ThL2d)*L2*sin(Th0+ThL0+ThL1+ThL2);
Adot(2,2) = Adot(2,3) - (Th0d+ThL1d)*L1*sin(Th0+ThL0+ThL1);
Adot(2,1) = Adot(2,2) - Th0d*L0*sin(Th0+ThL0);

```

```

Adot(1,3) = -(Th0d+ThL1d+ThL2d)*L2*cos(Th0+ThL0+ThL1+ThL2);
Adot(1,2) = Adot(1,3) - (Th0d+ThL1d)*L1*cos(Th0+ThL0+ThL1);
Adot(1,1) = Adot(1,2) - Th0d*L0*cos(Th0+ThL0);

```

```

%%%%%%%%%
%% B %%
%%%%%%%%%
B = zeros(8,6);
B(1,3) = -1;
B(1,6) = -1;
B(2,1) = 1;
B(2,3) = -1;
B(3,2) = 1;
B(3,3) = -1;
B(4,4) = 1;
B(4,6) = -1;
B(5,5) = 1;
B(5,6) = -1;
B(6,3) = 1;
B(6,6) = 1;

```

## J. MatxFix

```

% Filename is 'MatxFix.m'
% This routine calculates the matrices for the dual arm
% spacecraft EOM when it is grasping a payload. Each arm
% has two links. This version assumes that the centerbody
% is fixed. This impacts A and Adot.

function [M,G,A,Adot,B] = Matx(Ls,Ms,CMs,Is,Ths,Thdots,AngConst)

% OUTPUTS:
% M = 8x8 mass matrix
% G = 8x1 vector with coriolis and centripetal terms
% A = 5x8 constraints matrix
% Adot = 5x8 derivative of constraints matrix
% B = Control influence matrix
%
% INPUTS:
% Ls = 7x1 vector of lengths (m)
% 1st element = distance from origin to left arm mount
% 2nd & 3rd elements wrt left arm (from shoulder to wrist)
% 4th element = payload length
% 5th & 6th elements wrt right arm (from wrist to shoulder)
% 7th element = distance from right arm mount to origin
% [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)
% 1st element = mass of spacecraft centerbody
% 2nd & 3rd elements = left arm (upper then lower arm)
% 4th & 5th elements = right arm (upper then lower arm)
% 6th element = payload mass
% [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations (m)
% [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% Is = 6x1 column vector containing the moments of inertias
% about the respective body's center of mass (kg m^2)
% 1st element = inertia of spacecraft centerbody
% 2nd & 3rd elements = left arm (upper then lower arm)
% 4th & 5th elements = right arm (upper then lower arm)

```



```

%      6th element = payload inertia
%      [I0; IL1; IL2; IR1; IR2; IP]
% Ths = 6 element vector containing the angles which describe
%      the configuration of the system.
%      [Th0; ThL1; ThL2; ThR1; ThR2; ThP]
% Thdots = 6 element vector containing the angle rates
% AngConst = 2 element vector of arm mounting locations
%      [ThL0; ThR0]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lengths (m)
L0 = Ls(1);
L1 = Ls(2);
L2 = Ls(3);
LP = Ls(4);
R2 = Ls(5);
R1 = Ls(6);
R0 = Ls(7);

% Member masses (kg)
M0 = Ms(1);
ML1 = Ms(2);
ML2 = Ms(3);
MR1 = Ms(4);
MR2 = Ms(5);
MP = Ms(6);

% Center of mass distances (m)
Lc0 = CMs(1);
LcL1 = CMs(2);
LcL2 = CMs(3);
LcR1 = CMs(4);
LcR2 = CMs(5);
LcP = CMs(6); %measured from left end

% MOI about center of mass
I0 = Is(1);
IL1 = Is(2);
IL2 = Is(3);
IR1 = Is(4);
IR2 = Is(5);
IP = Is(6);

% Angles
Th0 = Ths(1);
ThL1 = Ths(2);
ThL2 = Ths(3);
ThR1 = Ths(4);
ThR2 = Ths(5);
ThP = Ths(6);

% Angle Rates
Th0d = Thdots(1);
ThL1d = Thdots(2);
ThL2d = Thdots(3);
ThR1d = Thdots(4);
ThR2d = Thdots(5);
ThPd = Thdots(6);

```

```

% Arm mount locations
ThL0 = AngConst(1);
ThR0 = AngConst(2);

%%%%%%%%%%
%% Mass Matrix %%
%%%%%%%%%%
M = zeros(8,8);
M(8,8) = MP;
M(7,7) = MP;
M(6,6) = IP;
M(5,5) = IR2 + MR2*LcR2^2;
M(5,4) = M(5,5) + MR2*R1*LcR2*cos(ThR2);
M(4,5) = M(5,4);
M(5,1) = M(4,5) + MR2*R0*LcR2*cos(ThR1+ThR2);
M(1,5) = M(5,1);
M(4,4) = M(4,5)+IR1+MR2*R1*LcR2*cos(ThR2)+MR1*LcR1^2+MR2*R1^2;
M(4,1)=M(4,4)+R0*(MR1*LcR1+MR2*R1)*cos(ThR1)+MR2*R0*LcR2*...
    cos(ThR1+ThR2);
M(1,4) = M(4,1);
M(3,3) = IL2 + ML2*LcL2^2;
M(3,2) = M(3,3) + ML2*L1*LcL2*cos(ThL2);
M(2,3) = M(3,2);
M(3,1) = M(3,2) + ML2*L0*LcL2*cos(ThL1+ThL2);
M(1,3) = M(3,1);
M(2,2) = M(3,2)+ML2*L1*LcL2*cos(ThL2)+IL1+ML1*LcL1^2+ML2*L1^2;
M(2,1)=M(2,2)+L0*(ML1*LcL1+ML2*L1)*cos(ThL1)+ML2*L0*LcL2*...
    cos(ThL1+ThL2);
M(1,2) = M(2,1);
Part1 = I0+M(2,2)+M(4,4)+M0*Lc0^2+(ML1+ML2)*L0^2+(MR1+MR2)*R0^2;
Part2 = 2*L0*(ML1*LcL1+ML2*L1)*cos(ThL1)+2*ML2*L0*LcL2*...
    cos(ThL1+ThL2);
Part3 = 2*R0*(MR1*LcR1+MR2*R1)*cos(ThR1)+2*MR2*R0*LcR2*...
    cos(ThR1+ThR2);
M(1,1) = Part1 + Part2 + Part3;

%%%%%%%%%%
%% G %%
%%%%%%%%%%
G = zeros(8,1);
Pt1 = -L0*(ThL1d^2+2*Th0d*ThL1d)*(ML1*LcL1+ML2*L1)*sin(ThL1);
Pt2 = -ML2*L1*LcL2*ThL2d*(2*Th0d+2*ThL1d+ThL2d)*sin(ThL2);
Pt3 = -ML2*L0*LcL2*(2*Th0d*(ThL1d+ThL2d)+(ThL1d+ThL2d)^2)*...
    sin(ThL1+ThL2);
Pt4 = -R0*(ThR1d^2+2*Th0d*ThR1d)*(MR1*LcR1+MR2*R1)*sin(ThR1);
Pt5 = -MR2*R1*LcR2*ThR2d*(2*Th0d+2*ThR1d+ThR2d)*sin(ThR2);
Pt6 = -MR2*R0*LcR2*(2*Th0d*(ThR1d+ThR2d)+(ThR1d+ThR2d)^2)*...
    sin(ThR1+ThR2);
G(1) = Pt1 + Pt2 + Pt3 + Pt4 + Pt5 + Pt6;
Pt1 = L0*Th0d^2*(ML1*LcL1+ML2*L1)*sin(ThL1);
Pt2 = -ML2*L1*LcL2*ThL2d*(2*Th0d+2*ThL1d+ThL2d)*sin(ThL2);
Pt3 = ML2*L0*LcL2*Th0d^2*sin(ThL1+ThL2);
G(2) = Pt1 + Pt2 + Pt3;
G(3) = ML2*LcL2*(L1*(Th0d+ThL1d)^2*sin(ThL2)+L0*Th0d^2*...
    sin(ThL1+ThL2));
Pt1 = R0*Th0d^2*(MR1*LcR1+MR2*R1)*sin(ThR1);
Pt2 = -MR2*R1*LcR2*ThR2d*(2*Th0d+2*ThR1d+ThR2d)*sin(ThR2);
Pt3 = MR2*R0*LcR2*Th0d^2*sin(ThR1+ThR2);
G(4) = Pt1 + Pt2 + Pt3;
G(5) = MR2*LcR2*(R1*(Th0d+ThR1d)^2*sin(ThR2)+R0*Th0d^2*...
    sin(ThR1+ThR2));

```

```

%%%%%%%%%%
%% Constraints Matrix %%
%%%%%%%%%%
% The constraint matrix comes from putting the constraint equations
% into the Pfaffian form:  $A \cdot \dot{q} + A_0 = 0$ . The first two constraint
% equations are found by finding the x and y components of the
% Payload's center of mass by starting at the origin and moving
% up the left arm. The second two constraint equations find the x
% and y components of the Payload's center of mass by starting at the
% origin and moving to the base of the right arm and then
% up the right arm. Differentiating these equations results
% in the Pfaffian form with  $A_0 = 0$ .
Lambda = zeros(5,8);
Lambda(5,1) = 1;
Lambda(1,7) = -1;
Lambda(2,8) = -1;
Lambda(3,7) = -1;
Lambda(4,8) = -1;
Lambda(1,6) = -LcP*sin(ThP);
Lambda(2,6) = LcP*cos(ThP);
Lambda(3,6) = (LP-LcP)*sin(ThP);
Lambda(4,6) = -(LP-LcP)*cos(ThP);
Lambda(4,5) = R2*cos(Th0+ThR0+ThR1+ThR2);
Lambda(4,4) = A(4,5) + R1*cos(Th0+ThR0+ThR1);
Lambda(4,1) = A(4,4) + R0*cos(Th0+ThR0);
Lambda(3,5) = -R2*sin(Th0+ThR0+ThR1+ThR2);
Lambda(3,4) = A(3,5) - R1*sin(Th0+ThR0+ThR1);
Lambda(3,1) = A(3,4) - R0*sin(Th0+ThR0);
Lambda(2,3) = L2*cos(Th0+ThL0+ThL1+ThL2);
Lambda(2,2) = A(2,3) + L1*cos(Th0+ThL0+ThL1);
Lambda(2,1) = A(2,2) + L0*cos(Th0+ThL0);
Lambda(1,3) = -L2*sin(Th0+ThL0+ThL1+ThL2);
Lambda(1,2) = A(1,3) - L1*sin(Th0+ThL0+ThL1);
Lambda(1,1) = A(1,2) - L0*sin(Th0+ThL0);

Adot = zeros(5,8);
Adot(1,6) = -ThPd*LcP*cos(ThP);
Adot(2,6) = -ThPd*LcP*sin(ThP);
Adot(3,6) = ThPd*(LP-LcP)*cos(ThP);
Adot(4,6) = ThPd*(LP-LcP)*sin(ThP);
Adot(4,5) = -(Th0d+ThR1d+ThR2d)*R2*sin(Th0+ThR0+ThR1+ThR2);
Adot(4,4) = Adot(4,5) - (Th0d+ThR1d)*R1*sin(Th0+ThR0+ThR1);
Adot(4,1) = Adot(4,4) - Th0d*R0*sin(Th0+ThR0);
Adot(3,5) = -(Th0d+ThR1d+ThR2d)*R2*cos(Th0+ThR0+ThR1+ThR2);
Adot(3,4) = Adot(3,5) - (Th0d+ThR1d)*R1*cos(Th0+ThR0+ThR1);
Adot(3,1) = Adot(3,4) - Th0d*R0*cos(Th0+ThR0);
Adot(2,3) = -(Th0d+ThL1d+ThL2d)*L2*sin(Th0+ThL0+ThL1+ThL2);
Adot(2,2) = Adot(2,3) - (Th0d+ThL1d)*L1*sin(Th0+ThL0+ThL1);
Adot(2,1) = Adot(2,2) - Th0d*L0*sin(Th0+ThL0);
Adot(1,3) = -(Th0d+ThL1d+ThL2d)*L2*cos(Th0+ThL0+ThL1+ThL2);
Adot(1,2) = Adot(1,3) - (Th0d+ThL1d)*L1*cos(Th0+ThL0+ThL1);
Adot(1,1) = Adot(1,2) - Th0d*L0*cos(Th0+ThL0);

%%%%%%%%%%
%% B %%
%%%%%%%%%%
B = zeros(8,6);
B(1,3) = -1;
B(1,6) = -1;
B(2,1) = 1;

```

```

B(2,3) = -1;
B(3,2) = 1;
B(3,3) = -1;
B(4,4) = 1;
B(4,6) = -1;
B(5,5) = 1;
B(5,6) = -1;
B(6,3) = 1;
B(6,6) = 1;

```

## K. Ref2

```

% Filename is 'Ref2.m'
% Reference Maneuver using cost function
% This routine assumes that the spacecraft centerbody is held fixed.

function [Torques,QRef,QdotRef,Aqdot,J,C1,C2,C3] = ...
    Ref2(Ls,Ms,CMs,Is,BoundC,T,Wu,Wc,Coef,ConstMat)

% OUTPUTS:
% Torques = 7x1 column vector of torques that should be applied at
%         time T if the motion is to follow the reference trajectory
%         exactly. The vector is arranged as [U0; ULS; ULE; ULW; URS; URE; URW]
%         which are the centerbody torque followed by the torques at the
%         shoulder, elbow, and wrist of the left arm and then the right arm
%         respectively.
% QRef = 8x1 column vector of reference generalized coordinates
% QdotRef = 8x1 column vector of reference generalized velocities
% Aqdot = 4x1 or 5x1 column vector (depends on status of AMatFlag) which
%         check to see if the constraint equation A*Qdot = 0 is satisfied
% J = scalar value of the reaction wheel torque absolute value. This
%     number will be integrated to find the value for the cost function.
% Lyapunov Controller matrices (reference trajectory values)
%     C1 = 8x7 matrix
%     C2 = 8x4 or 8x5 (depends on status of AMatFlag) matrix
%     C3 = 8x1 matrix
%
% INPUTS:
% Ls = 7x1 vector of lengths (m)
%     1st element = distance from origin to left arm mount
%     2nd & 3rd elements wrt left arm (from shoulder toward wrist)
%     4th element = payload length
%     5th & 6th elements wrt right arm (from wrist toward shoulder)
%     7th element = distance from right arm mount to origin
%     [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)
%     1st element = mass of spacecraft centerbody
%     2nd & 3rd elements = mass of left arm (upper arm then lower arm)
%     4th & 5th elements = mass of right arm (upper arm then lower arm)
%     6th element = payload mass
%     [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations
%     [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% Is = 6x1 column vector containing the moments of inertias about the
%     respective body's center of mass (kg m^2)
%     1st element = inertia of spacecraft centerbody
%     2nd & 3rd elements = inertia of left arm (upper arm then lower arm)
%     4th & 5th elements = inertia of right arm (upper arm then lower arm)
%     6th element = payload inertia

```

```

%      [I0; IL1; IL2; IR1; IR2; IP]
% BoundC = boundry conditions for the problem. The first column
%      contains the initial x and y component of points Q & P
%      respectively, the x component of the right arm base, the
%      problem start time, and the simulation stop time. The second
%      column contains the x and y component of points Q & P
%      respectively, the x component of the right arm base, the
%      stop time for the ideal reference maneuver, and a flag to
%      activate or deactivate the controller. The origin for the
%      x and y components is the base of the left arm.
% T = time
% Wu = 6x6 or 7x7 control torque cost weighting matrix
% Wc = 8x8 constraint cost weighting matrix
% Coef = (n-2)x1 column vector of reference polynomial coefficients
%      beginning with order n coefficient
% ConstMat = 3x(n-2) matrix of coefficients for reference displacement
%      (row 1), velocity (row 2), and acceleration (row 3)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lengths (m)
L0 = Ls(1);
L1 = Ls(2);
L2 = Ls(3);
LP = Ls(4);
R2 = Ls(5);
R1 = Ls(6);
R0 = Ls(7);

% Member masses (kg)
M0 = Ms(1);
ML1 = Ms(2);
ML2 = Ms(3);
MR1 = Ms(4);
MR2 = Ms(5);
MP = Ms(6);

% Center of mass distances (m)
Lc0 = CMs(1);
LcL1 = CMs(2);
LcL2 = CMs(3);
LcR1 = CMs(4);
LcR2 = CMs(5);
LcP = CMs(6); %measured from left end

% MOI about center of mass
I0 = Is(1);
IL1 = Is(2);
IL2 = Is(3);
IR1 = Is(4);
IR2 = Is(5);
IP = Is(6);

% Initial and final locations of third link
% Point Q is at Node 3 (joint between Links 2 & 3)
% Point P is at Node 4 (joint between Links 3 & 4)
Qx0 = BoundC(1,1); Qy0 = BoundC(1,2);
Px0 = BoundC(2,1); Py0 = BoundC(2,2);
Qxf = BoundC(3,1); Qyf = BoundC(3,2);
Pxf = BoundC(4,1); Pyf = BoundC(4,2);

```

```

% Arms mount locations wrt spacecraft centerbody coordinate frame (rad)
ThI,0 = BoundC(5,1); ThR0 = BoundC(5,2);

% Reference Maneuver Start and Stop Times
T0 = BoundC(6,1); Tf = BoundC(6,2);

% Constraints Matrix Flag
AMatFlag = BoundC(8,1);

% Centerbody Reaction Wheel Flag
WheelFlag = BoundC(8,2);

% Centerbody Initial and Final Conditions
Th00 = BoundC(9,1);
Th0f = BoundC(9,2);

% Number of equations in the cost function constraint equations
EOMFlag = BoundC(10,1);

% Psuedo-Inverse Flag
PInvFlag = BoundC(10,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PRELIMINARY CALCULATIONS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R2D = 180/pi; % Conversion from radians to degrees

% Total rotation of Payload
ThP0 = atan2(Py0-Qy0,Px0-Qx0); % Initial angle of Payload (rad)
ThPf = atan2(Pyf-Qyf,Pxf-Qxf); % Final angle of Payload (rad)
DelThP = ThPf - ThP0; % Total delta angle of Payload (rad)

% Initial and final locations of Payload center of mass
XP0 = Qx0 + (Px0 - Qx0) * (LcP/LP);
YP0 = Qy0 + (Py0 - Qy0) * (LcP/LP);
XPf = Qxf + (Pxf - Qxf) * (LcP/LP);
YPf = Qyf + (Pyf - Qyf) * (LcP/LP);

Tau = (T-T0) / (Tf-T0); % Normalize time

% Function Weighting Factors for how the payload will move
% These factors will cause the velocity and acceleration of
% the payload coordinates to be zero at t = 0 and t = tf.
% They also permit the displacements for the payload coordinates
% to match their initial and final values. These weighting
% factors will also apply to the centerbody rotation.

k = length(Coef);
for n=1:k
    CTau(k+1-n) = Coef(k+1-n)*Tau^(n+2);
    CTaud(k+1-n) = Coef(k+1-n)*Tau^(n+1);
    CTaudd(k+1-n) = Coef(k+1-n)*Tau^(n);
end
% Weighting factors
W = ConstMat(1,:)*CTau';
Wd = ConstMat(2,:)*CTaud';
Wdd = ConstMat(3,:)*CTaudd';

% Centerbody angle, angular velocity, angular acceleration
DelTh0 = Th0f - Th00;

```

```

Th0 = Th00 + W * DelTh0;           % Angle (rad);
Th0d = Wd * DelTh0 / (Tf - T0);    % Velocity (rad/sec);
Th0dd = Wdd * DelTh0 / (Tf - T0)^2; % Acceleration (rad/sec^2);
%Th0 = 0;
%Th0d = 0;
%Th0dd = 0;
% Save for plotting
QRef(1) = Th0;
QdotRef(1) = Th0d;
QddotRef(1) = Th0dd;

% Payload angle, angular velocity, angular acceleration
ThP = ThP0 + W * DelThP;           % Angle (rad)
ThPd = Wd * DelThP / (Tf - T0);    % Velocity (rad/sec)
ThPdd = Wdd * DelThP / (Tf - T0)^2; % Acceleration (rad/sec^2)
% Save for plotting
QRef(6) = ThP;
QdotRef(6) = ThPd;
QddotRef(6) = ThPdd;

% Payload center of mass position, velocity, and acceleration
Xc = XP0 + W * (XPf - XP0);
Xcd = Wd * (XPf - XP0) / (Tf - T0);
Xcdd = Wdd * (XPf - XP0) / (Tf - T0)^2;
Yc = YP0 + W * (YPf - YP0);
Ycd = Wd * (YPf - YP0) / (Tf - T0);
Ycdd = Wdd * (YPf - YP0) / (Tf - T0)^2;
% Save for plotting
QRef(7) = Xc;
QdotRef(7) = Xcd;
QddotRef(7) = Xcdd;
QRef(8) = Yc;
QdotRef(8) = Ycd;
QddotRef(8) = Ycdd;

% Payload endpoint coordinates: Qx, Qy, Px, Py
Qx = Xc - LcP * cos(ThP);
Qy = Yc - LcP * sin(ThP);
Px = Xc + (LP - LcP) * cos(ThP);
Py = Yc + (LP - LcP) * sin(ThP);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solve for Arm Angles Required by desired path %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LEFT ARM %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Elbow is left of line from arm base to Q (RQ)
LSx = L0 * cos(Th0 + ThL0);
LSy = L0 * sin(Th0 + ThL0);
RQ = sqrt((Qx-LSx)^2+(Qy-LSy)^2); % Length from arm base to Q
Beta1 = atan2(Qy-LSy,Qx-LSx);    % Angle from arm base to RQ
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RQ and Link L1
Num = L1^2 + RQ^2 - L2^2;
Den = 2 * L1 * RQ;
Beta2 = acos(Nom/Den);           % Angle from RQ to Link 1
ThL1 = (Beta1 + Beta2) - (Th0 + ThL0); % Theta L1
% Use law of cosines to find the interior angle at the elbow
Num = L1^2 + L2^2 - RQ^2;
Den = 2 * L1 * L2;

```

```

Beta3 = acos(Num/Den);
ThL2 = -(pi-Beta3);
%Save for plotting
QRef(2) = ThL1;
QRef(3) = ThL2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% RIGHT ARM %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Elbow is right of line from arm base (shoulder) to P (wrist) (RP)
RSx = R0 * cos(Th0 + ThR0);
RSy = R0 * sin(Th0 + ThR0);
RP = sqrt((Px-RSx)^2+(Py-RSy)^2); % Length from arm base to P
Beta1 = atan2(Py-RSy,Px-RSx); % Angle from arm base to RP
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RP and Link R1
Num = R1^2 + RP^2 - R2^2;
Den = 2 * R1 * RP;
Beta2 = acos(Num/Den); % Angle from Link R1 to RP
Beta4 = Beta1 - (Th0 + ThR0);
ThR1 = -(Beta2 - Beta4);
Num = R1^2 + R2^2 - RP^2;
Den = 2 * R1 * R2;
Beta3 = acos(Num/Den);
ThR2 = pi - Beta3;
% Save for plotting
QRef(4) = ThR1;
QRef(5) = ThR2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solve for Arm Angle Rates & Accelerations required by desired path %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LEFT ARM %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd
% Qxd & Qyd are x & y components of point Q inertial velocity.
% Thd = [ThL1dot; ThL2dot]
% H matrices are made from expressing the x & y components of Q in
% terms of L0, Th0, ThL0, L1, ThL1, L2, and ThL2.
% Qx=L0*cos(Th0+ThL0)+L1*cos(Th0+ThL0+ThL1)+L2*cos(Th0+...
% ThL0+ThL1+ThL2)
% Qy=L0*sin(Th0+ThL0)+L1*sin(Th0+ThL0+ThL1)+L2*sin(Th0+...
% ThL0+ThL1+ThL2)
% The differentiation of these equations lead to
% [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd which can be solved for Thd
Qxd = Xcd + LcP * ThPd * sin(ThP);
Qyd = Ycd - LcP * ThPd * cos(ThP);
H2(1,2) = -L2*sin(Th0+ThL0+ThL1+ThL2);
H2(1,1) = H2(1,2) - L1*sin(Th0+ThL0+ThL1);
H2(2,2) = L2*cos(Th0+ThL0+ThL1+ThL2);
H2(2,1) = H2(2,2) + L1*cos(Th0+ThL0+ThL1);
H1(1,1) = H2(1,1) - L0*sin(Th0+ThL0);
H1(2,1) = H2(2,1) + L0*cos(Th0+ThL0);
Thd = inv(H2) * ([Qxd; Qyd] - H1*Th0d);
% Angle rates
ThL1d = Thd(1);
ThL2d = Thd(2);
% Save for plotting
QdotRef(2) = ThL1d;
QdotRef(3) = ThL2d;

```



```

% Differentiation of [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd leads to
% [Qxdd; Qydd] = [H1dot]*Th0d+[H1]*Th0dd + [H2dot]*Thd+[H2]*Thdd
Qxdd = Xcdd + LcP*(ThPdd*sin(ThP) + ThPd^2*cos(ThP));
Qydd = Ycdd - LcP*(ThPdd*cos(ThP) - ThPd^2*sin(ThP));
H2dot(1,2) = -L2*(Th0d+ThL1d+ThL2d)*cos(Th0+ThL0+ThL1+ThL2);
H2dot(1,1) = H2dot(1,2) - L1*(Th0d+ThL1d)*cos(Th0+ThL0+ThL1);
H2dot(2,2) = -L2*(Th0d+ThL1d+ThL2d)*sin(Th0+ThL0+ThL1+ThL2);
H2dot(2,1) = H2dot(2,2) - L1*(Th0d+ThL1d)*sin(Th0+ThL0+ThL1);
H1dot(1,1) = H2dot(1,1) - L0*Th0d*cos(Th0+ThL0);
H1dot(2,1) = H2dot(2,1) - L0*Th0d*sin(Th0+ThL0);
Thdd = inv(H2)*([Qxdd; Qydd]-H2dot*Thd-[H1dot]*Th0d-[H1]*Th0dd);
% Angle accelerations
ThL1dd = Thdd(1);
ThL2dd = Thdd(2);
QddotRef(2) = ThL1dd;
QddotRef(3) = ThL2dd;

%%%%%%%%%%%%%%
%% RIGHT ARM %%
%%%%%%%%%%%%%%
% The development is similar to the left arm
% Px=R0*cos(Th0+ThR0)+R1*cos(Th0+ThR0+ThR1)+R2*cos(Th0+...
%   ThR0+ThR1+ThR2)
% Py=R0*sin(Th0+ThR0)+R1*sin(Th0+ThR0+ThR1)+R2*sin(Th0+...
%   ThR0+ThR1+ThR2)
% [Pxd; Pyd] = [H1]*Th0d + [H2]*Thd
Pxd = Xcd - (LP - LcP) * ThPd * sin(ThP);
Pyd = Ycd + (LP - LcP) * ThPd * cos(ThP);
H2(1,2) = -R2*sin(Th0+ThR0+ThR1+ThR2);
H2(1,1) = H2(1,2) - R1*sin(Th0+ThR0+ThR1);
H2(2,2) = R2*cos(Th0+ThR0+ThR1+ThR2);
H2(2,1) = H2(2,2) + R1*cos(Th0+ThR0+ThR1);
H1(1,1) = H2(1,1) - R0*sin(Th0+ThR0);
H1(2,1) = H2(2,1) + R0*cos(Th0+ThR0);
Thd = inv(H2) * ([Pxd; Pyd] - H1*Th0d);
% Angle rates
ThR1d = Thd(1);
ThR2d = Thd(2);
% Save for plotting
QdotRef(4) = ThR1d;
QdotRef(5) = ThR2d;

% [Pxdd; Pydd] = [H1dot]*Th0d+[H1]*Th0dd + [H2dot]*Thd+[H2]*Thdd
Pxdd = Xcdd - (LP - LcP)*(ThPdd*sin(ThP) + ThPd^2*cos(ThP));
Pydd = Ycdd + (LP - LcP)*(ThPdd*cos(ThP) - ThPd^2*sin(ThP));
H2dot(1,2) = -R2*(Th0d+ThR1d+ThR2d)*cos(Th0+ThR0+ThR1+ThR2);
H2dot(1,1) = H2dot(1,2) - R1*(Th0d+ThR1d)*cos(Th0+ThR0+ThR1);
H2dot(2,2) = -R2*(Th0d+ThR1d+ThR2d)*sin(Th0+ThR0+ThR1+ThR2);
H2dot(2,1) = H2dot(2,2) - R1*(Th0d+ThR1d)*sin(Th0+ThR0+ThR1);
H1dot(1,1) = H2dot(1,1) - R0*Th0d*cos(Th0+ThR0);
H1dot(2,1) = H2dot(2,1) - R0*Th0d*sin(Th0+ThR0);
Thdd = inv(H2)*([Pxdd; Pydd]-H2dot*Thd-[H1dot]*Th0d-[H1]*Th0dd);
% Angle accelerations
ThR1dd = Thdd(1);
ThR2dd = Thdd(2);
QddotRef(4) = ThR1dd;
QddotRef(5) = ThR2dd;

%%%%%%%%%%%%%%
%% Find needed control torques, u %%

```

```

%%%%%%%%%%
% EOM:      M*qddot + G = B*u + A'*Lam
% Constraint Eqns: A*qdot = 0
% Solve EOM for qddot leads to
% qddot = inv(M)*(B*u + A'*Lam - G)
% Differentiate Constraint Eqns gives Adot*qdot + A*qddot = 0
% Substitute qddot derived from EOM into differentiated constraint
% eqns and solve for Lam
% Lam = -inv(A*inv(M)*A')*(A*inv(M)*(B*u-G)+Adot*qdot)
% Substitute this expression for Lam into original EOM. Collect terms
% into the form MTilda*qddot + GTilda = BTilda*u
% where MTilda = M
% GTilda = G + A'*inv(A*inv(M)*A')*(Adot*qdot-A*inv(M)*G)
% BTilda = (I-A'*inv(A*inv(M)*A')*A*inv(M))*B
% The first five resulting equations apply to the spacecraft centerbody
% and arms. The final three apply to the payload. The matrix form of
% the last three equations is
% MPTilda*QPddot + GPTilda = BPTilda*u

%%%%%%%%%%
%% Matrices %%
%%%%%%%%%%
AngConst = [ThL0; ThR0];
%AMatFlag = 1;
if AMatFlag
    [M,G,A,Adot,B] = MatxFix(Ls,Ms,CMs,Is,QRef,QdotRef,AngConst);
else
    [M,G,A,Adot,B] = Matx(Ls,Ms,CMs,Is,QRef,QdotRef,AngConst);
end

if WheelFlag
    B7 = [1; 0; 0; 0; 0; 0; 0; 0];
    B = [B7 B];
end

% If the cost function is subject to the constraint that the payload
% satisfy the reference motion, then three equations of motion are used.
% To include the centerbody reference motion, use four equations from
% the equations of motion.

%%%%%%%%%%
%% MTilda %%
%%%%%%%%%%
if EOMFlag == 3 % Use only the payload equations
    MPTilda = M(6:8,6:8);
else
    if EOMFlag == 5 % Use the spacecraft equations
        MPTilda = M(1:5,1:5);
    else % Use all eight equations
        MPTilda = M;
    end
end

%%%%%%%%%%
%% GTilda %%
%%%%%%%%%%
Qdot = QdotRef;
GTilda = G + A'*inv(A*inv(M)*A')*(Adot*Qdot - A*inv(M)*G);
if EOMFlag == 3 % Use only the payload equations
    GPTilda = GTilda(6:8,1);
else

```

```

if EOMFlag == 5 % Use the spacecraft equations
    GPTilda = GTilda(1:5,1);
else % Use all eight equations
    GPTilda = GTilda;
end
end

%%%%%%%%%%%%
%% BTilda %%
%%%%%%%%%%%%
BTilda = (eye(8) - A'*inv(A*inv(M)*A')*A*inv(M))*B;
if EOMFlag == 3 % Use only the payload equations
    BPTilda = BTilda(6:8,:);
else
    if EOMFlag == 5 % Use the spacecraft equations
        BPTilda = BTilda(1:5,:);
    else % Use all eight equations
        BPTilda = BTilda;
    end
end

%%%%%%%%%%%%
%% G1 & G2 %%
%%%%%%%%%%%%
% Use previous expression for Lam and regroup terms into the following
% form A'*Lam = K1 + K2*u
K1 = A'*inv(A*inv(M)*A')*(A*inv(M)*G-Adot*Qdot);
K2 = -A'*inv(A*inv(M)*A')*A*inv(M)*B;

%%%%%%%%%%%%
%% Torques %%
%%%%%%%%%%%%
% Torques are calculated to minimize the following cost function:
% J = 0.5*[u'*Wu*u + Lam'*A*Wc*A'*Lam + Tr*Wr*Tr]
% Subject to the constraint: MP*QPddot + GPTilda - BPTilda*u = 0
% Combine the constraint into the cost function by multiplying the
% constraint eqn by another Lagrange multiplier, Gam, and adding that
% to the cost function. Take the gradient with respect to u results in
% (Wu+K2'*Wc*K2)*u + K2'*Wc*K1 - BPTilda'*Gam = 0
% Solve for u
% u = inv(Wu+K2'*Wc*K2)*(BPTilda'*Gam - K2'*Wc*K1)
% Substitute this into the constraint eqn. Solve the result for Gam
% Gam = inv(BPTilda*inv(Wu + K2'*Wc*K2)*BPTilda')*(MP*QPddot+
% GPTilda+BPTilda*inv(Wu + K2'*Wc*K2)*K2'*Wc*K1)
% Substitute this expression into the torque equation, u.

Qddot = QddotRef;

if EOMFlag == 3 % Use only the payload equations
    QPddot = Qddot(6:8,1);
else
    if EOMFlag == 5 % Use the spacecraft equations
        QPddot = Qddot(1:5,1);
    else % Use all eight equations
        QPddot = Qddot;
    end
end

%%%%%%%%%%%%
%% PSUEDO-INVERSES %%
%%%%%%%%%%%%

```

```

% To avoid the problems with poorly conditioned matrices, I've used the
% psuedo-inverse rather than the traditional inverse in the next two
% equations.
if PInvFlag
    Part1 = pinv(Wu + K2'*Wc*K2);
    Gam = pinv(BPTilda*Part1*BPTilda') * (MPTilda*QPddot + GPTilda + ...
        BPTilda*Part1*(K2'*Wc*K1));
else
    Part1 = inv(Wu + K2'*Wc*K2);
    Gam = inv(BPTilda*Part1*BPTilda') * (MPTilda*QPddot + GPTilda + ...
        BPTilda*Part1*(K2'*Wc*K1));
end

% Reference Torques
Torques = Part1*(BPTilda*Gam - K2'*Wc*K1);

% Cost Function, J
J = abs(Torques(1));

%Controller Info
Pt1 = A'*inv(A*inv(M)*A');
C1 = inv(M)*(eye(M) - Pt1*A*inv(M))*B;
C2 = -inv(M)*Pt1*Adot;
C3 = inv(M)*(Pt1*A*inv(M) - eye(M))*G;

%%%%%%%%%%
%%% DEBUG INFO %%%
%%%%%%%%%%
%%% Are constraint equations, A*qdot=0, satisfied?
%qdot = A*QdotRef;

```

## L. RefMin2

```

% Filename is 'RefMin2.m'
% Reference Maneuver using cost function
% This routine is used by "MainOpt.m" to find the optimal combination
% of reference trajectory polynomial coefficients.
% Version 2 uses the rate of change of angular momentum to find
% the wheel torque.

function [Jopt1,Jopt2] = RefMin2(T,Ls,Ms,CMs,Is,BoundC,Wu,Wc,Coef,ConstMat)

% OUTPUTS:
% Jopt = absolute value of the reaction wheel torque. This is the cost
% function value for purposes of optimizing the reference
% trajectory polynomial coefficients. Jopt1 will be integrated by
% odemin.m while Jopt2 is the same value but won't be integrated.
%
% INPUTS:
% T = time (sec)
% Ls = 7x1 vector of lengths (m)
% 1st element = distance from origin to left arm mount
% 2nd & 3rd elements wrt left arm (from shoulder toward wrist)
% 4th element = payload length
% 5th & 6th elements wrt right arm (from wrist toward shoulder)
% 7th element = distance from right arm mount to origin
% [L0; L1; L2; LP; R2; R1; R0]
% Ms = 6x1 column vector containing the masses (kg)
% 1st element = mass of spacecraft centerbody

```

```

%      2nd & 3rd elements = mass of left arm (upper arm then lower arm)
%      4th & 5th elements = mass of right arm (upper arm then lower arm)
%      6th element = payload mass
%      [M0; ML1; ML2; MR1; MR2; MP]
% CMs = 6x1 column vector containing center of mass locations
%      [Lc0; LcL1; LcL2; LcR1; LcR2; LcP]
% ls = 6x1 column vector containing the moments of inertias about the
%      respective body's center of mass (kg m^2)
%      1st element = inertia of spacecraft centerbody
%      2nd & 3rd elements = inertia of left arm (upper arm then lower arm)
%      4th & 5th elements = inertia of right arm (upper arm then lower arm)
%      6th element = payload inertia
%      [I0; IL1; IL2; IR1; IR2; IP]
% BoundC = boundry conditions for the problem. The first column
%      contains the initial x and y component of points Q & P
%      respectively, the x component of the right arm base, the
%      problem start time, and the simulation stop time. The second
%      column contains the x and y component of points Q & P
%      respectively, the x component of the right arm base, the
%      stop time for the ideal reference maneuver, and a flag to
%      activate or deactivate the controller. The origin for the
%      x and y components is the base of the left arm.
% Wu = 6x6 control torque cost weighting matrix
% Wc = 8x8 constraint cost weighting matrix
% Coef = (n-2)x1 vector of polynomial reference trajectory coefficients
%      in descending order where n is the highest order coefficient
% ConstMat = 3x(n-2) matrix of coefficients for reference trajectory
%      displacement (row 1), velocity (row 2) and acceleration (row 3)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CONVERT INPUTS FROM ARRAYS TO SCALARS %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Lengths (m)
L0 = Ls(1);
L1 = Ls(2);
L2 = Ls(3);
LP = Ls(4);
R2 = Ls(5);
R1 = Ls(6);
R0 = Ls(7);

% Member masses (kg)
M0 = Ms(1);
ML1 = Ms(2);
ML2 = Ms(3);
MR1 = Ms(4);
MR2 = Ms(5);
MP = Ms(6);

% Center of mass distances (m)
Lc0 = CMs(1);
LcL1 = CMs(2);
LcL2 = CMs(3);
LcR1 = CMs(4);
LcR2 = CMs(5);
LcP = CMs(6); %measured from left end

% MOI about center of mass
I0 = Is(1);
IL1 = Is(2);
IL2 = Is(3);

```

```

IR1 = Is(4);
IR2 = Is(5);
IP = Is(6);

% Initial and final locations of third link
% Point Q is at Node 3 (joint between Links 2 & 3)
% Point P is at Node 4 (joint between Links 3 & 4)
Qx0 = BoundC(1,1); Qy0 = BoundC(1,2);
Px0 = BoundC(2,1); Py0 = BoundC(2,2);
Qxf = BoundC(3,1); Qyf = BoundC(3,2);
Pxf = BoundC(4,1); Pyf = BoundC(4,2);

% Arms mount locations wrt spacecraft centerbody coordinate frame (rad)
ThL0 = BoundC(5,1); hR0 = BoundC(5,2);

% Reference Maneuver Start and Stop Times
T0 = BoundC(6,1); Tf = BoundC(6,2);

% Constraints Matrix Flag
AMatFlag = BoundC(8,1);

% Centerbody Reaction Wheel Flag
WheelFlag = BoundC(8,2);

% Centerbody Initial and Final Conditions
Th00 = BoundC(9,1);
Th0f = BoundC(9,2);

% Number of equations in the cost function constraint equations
EOMFlag = BoundC(10,1);

% Psuedo-Inverse Flag
PInvFlag = BoundC(10,2);

%%%%%%%%%%
%% PRELIMINARY CALCULATIONS %%
%%%%%%%%%%
R2D = 180/pi; % Conversion from radians to degrees

% Total rotation of Payload
ThP0 = atan2(Py0-Qy0,Px0-Qx0); % Initial angle of Payload (rad)
ThPf = atan2(Pyf-Qyf,Pxf-Qxf); % Final angle of Payload (rad)
DelThP = ThPf - ThP0; % Total delta angle of Payload (rad)

% Initial and final locations of Payload center of mass
XP0 = Qx0 + (Px0 - Qx0) * (LcP/LP);
YP0 = Qy0 + (Py0 - Qy0) * (LcP/LP);
XPf = Qxf + (Pxf - Qxf) * (LcP/LP);
YPf = Qyf + (Pyf - Qyf) * (LcP/LP);

Tau = (T-T0) / (Tf-T0); % Normalize time

% Function Weighting Factors for how the payload will move
% These factors will cause the angular velocity and angular
% acceleration of the payload to be zero at t = 0 and t = tf
% They also permit the payload angle to match its initial
% and final values. These weighting factors will also apply
% to the translational motion of the payload center of mass.

k = length(Coef);
for n=1:k

```

```

CTau(k+1-n) = Coef(k+1-n)*Tau^(n+2);
CTaud(k+1-n) = Coef(k+1-n)*Tau^(n+1);
CTaudd(k+1-n) = Coef(k+1-n)*Tau^(n);
end
W = ConstMat(1,:)*CTau';
Wd = ConstMat(2,:)*CTaud';
Wdd = ConstMat(3,:)*CTaudd';

% Centerbody angle, angular velocity, angular acceleration
DelTh0 = Th0f - Th00;
Th0 = Th00 + W * DelTh0; % Angle (rad);
Th0d = Wd * DelTh0 / (Tf - T0); % Velocity (rad/sec);
Th0dd = Wdd * DelTh0 / (Tf - T0)^2; % Acceleration (rad/sec^2);
% Save for plotting
QRef(1) = Th0;
QdotRef(1) = Th0d;
QddotRef(1) = Th0dd;

% Payload angle, angular velocity, angular acceleration
ThP = ThP0 + W * DelThP; % Angle (rad)
ThPd = Wd * DelThP / (Tf - T0); % Velocity (rad/sec)
ThPdd = Wdd * DelThP / (Tf - T0)^2; % Acceleration (rad/sec^2)
% Save for plotting
QRef(6) = ThP;
QdotRef(6) = ThPd;
QddotRef(6) = ThPdd;

% Payload center of mass position, velocity, and acceleration
Xc = XP0 + W * (XPf - XP0);
Xcd = Wd * (XPf - XP0) / (Tf - T0);
Xcdd = Wdd * (XPf - XP0) / (Tf - T0)^2;
Yc = YP0 + W * (YPf - YP0);
Ycd = Wd * (YPf - YP0) / (Tf - T0);
Ycdd = Wdd * (YPf - YP0) / (Tf - T0)^2;
% Save for plotting
QRef(7) = Xc;
QdotRef(7) = Xcd;
QddotRef(7) = Xcdd;
QRef(8) = Yc;
QdotRef(8) = Ycd;
QddotRef(8) = Ycdd;

% Payload endpoint coordinates: Qx, Qy, Px, Py
Qx = Xc - LcP * cos(ThP);
Qy = Yc - LcP * sin(ThP);
Px = Xc + (LP - LcP) * cos(ThP);
Py = Yc + (LP - LcP) * sin(ThP);

%%%%%%%%%%%%%%
%% Solve for Arm Angles Required by desired path %%
%%%%%%%%%%%%%%
%% LEFT ARM %%
%%%%%%%%%%%%%%
% Elbow is left of line from arm base to Q (RQ)
LSx = L0 * cos(Th0 + ThL0);
LSy = L0 * sin(Th0 + ThL0);
RQ = sqrt((Qx-LSx)^2+(Qy-LSy)^2); % Length from arm base to Q
Beta1 = atan2(Qy-LSy,Qx-LSx); % Angle from arm base to RQ
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RQ and Link L1

```

```

Num = L1^2 + RQ^2 - L2^2;
Den = 2 * L1 * RQ;
Beta2 = acos(Num/Den); % Angle from RQ to Link 1
ThL1 = (Beta1 + Beta2) - (Th0 + ThL0); % Theta L1
% Use law of cosines to find the interior angle at the elbow
Num = L1^2 + L2^2 - RQ^2;
Den = 2 * L1 * L2;
Beta3 = acos(Num/Den);
ThL2 = -(pi-Beta3);
% Save for plotting
QRef(2) = ThL1;
QRef(3) = ThL2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% RIGHT ARM %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Elbow is right of line from arm base (shoulder) to P (wrist) (RP)
RSx = R0 * cos(Th0 + ThR0);
RSy = R0 * sin(Th0 + ThR0);
RP = sqrt((Px-RSx)^2+(Py-RSy)^2); % Length from arm base to P
Beta1 = atan2(Py-RSy,Px-RSx); % Angle from arm base to RP
% Law of cosines: cos(A) = (b^2 + c^2 - a^2)/(2bc)
% Apply to find angle between RP and Link R1
Num = R1^2 + RP^2 - R2^2;
Den = 2 * R1 * RP;
Beta2 = acos(Num/Den); % Angle from Link R1 to RP
Beta4 = Beta1 - (Th0 + ThR0);
ThR1 = -(Beta2 - Beta4);
Num = R1^2 + R2^2 - RP^2;
Den = 2 * R1 * R2;
Beta3 = acos(Num/Den);
ThR2 = pi - Beta3;
% Save for plotting
QRef(4) = ThR1;
QRef(5) = ThR2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Solve for Arm Angle Rates & Accelerations %%
%% required by desired path %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LEFT ARM %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd
% Qxd & Qyd are x & y components of point Q inertial velocity.
% Thd = [ThL1dot; ThL2dot]
% H matrices are made from expressing the x & y components of Q in
% terms of L0, Th0, ThL0, L1, ThL1, L2, and ThL2.
% Qx=L0*cos(Th0+ThL0)+L1*cos(Th0+ThL0+ThL1)+L2*cos(Th0+...
% ThL0+ThL1+ThL2)
% Qy=L0*sin(Th0+ThL0)+L1*sin(Th0+ThL0+ThL1)+L2*sin(Th0+...
% ThL0+ThL1+ThL2)
% The differentiation of these equations lead to
% [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd which can be solved for Thd
Qxd = Xcd + LcP * ThPd * sin(ThP);
Qyd = Ycd - LcP * ThPd * cos(ThP);
H2(1,2) = -L2*sin(Th0+ThL0+ThL1+ThL2);
H2(1,1) = H2(1,2) - L1*sin(Th0+ThL0+ThL1);
H2(2,2) = L2*cos(Th0+ThL0+ThL1+ThL2);
H2(2,1) = H2(2,2) + L1*cos(Th0+ThL0+ThL1);
H1(1,1) = H2(1,1) - L0*sin(Th0+ThL0);

```



```

H1(2,1) = H2(2,1) + L0*cos(Th0+ThL0);
Thd = inv(H2) * ([Qxd; Qyd] - H1*Th0d);
% Angle rates
ThL1d = Thd(1);
ThL2d = Thd(2);
% Save for plotting
QdotRef(2) = ThL1d;
QdotRef(3) = ThL2d;

% Differentiation of [Qxd; Qyd] = [H1]*Th0d + [H2]*Thd leads to
% [Qxdd; Qydd] = [H1dot]*Th0d+[H1]*Th0dd + [H2dot]*Thd+[H2]*Thdd
Qxdd = Xcdd + LcP*(ThPdd*sin(ThP) + ThPd^2*cos(ThP));
Qydd = Ycdd - LcP*(ThPdd*cos(ThP) - ThPd^2*sin(ThP));
H2dot(1,2) = -L2*(Th0d+ThL1d+ThL2d)*cos(Th0+ThL0+ThL1+ThL2);
H2dot(1,1) = H2dot(1,2) - L1*(Th0d+ThL1d)*cos(Th0+ThL0+ThL1);
H2dot(2,2) = -L2*(Th0d+ThL1d+ThL2d)*sin(Th0+ThL0+ThL1+ThL2);
H2dot(2,1) = H2dot(2,2) - L1*(Th0d+ThL1d)*sin(Th0+ThL0+ThL1);
H1dot(1,1) = H2dot(1,1) - L0*Th0d*cos(Th0+ThL0);
H1dot(2,1) = H2dot(2,1) - L0*Th0d*sin(Th0+ThL0);
Thdd = inv(H2)*([Qxdd; Qydd]-H2dot*Thd-[H1dot]*Th0d-[H1]*Th0dd);
% Angle accelerations
ThL1dd = Thdd(1);
ThL2dd = Thdd(2);
QddotRef(2) = ThL1dd;
QddotRef(3) = ThL2dd;

%%%%%%%%%%
%% RIGHT ARM %%
%%%%%%%%%%
% The development is similar to the left arm
% Px=R0*cos(Th0+ThR0)+R1*cos(Th0+ThR0+ThR1)+R2*cos(Th0+...
% ThR0+ThR1+ThR2)
% Py=R0*sin(Th0+ThR0)+R1*sin(Th0+ThR0+ThR1)+R2*sin(Th0+...
% ThR0+ThR1+ThR2)
% [Pxd; Pyd] = [H1]*Th0d + [H2]*Thd
Pxd = Xcd - (LP - LcP) * ThPd * sin(ThP);
Pyd = Ycd + (LP - LcP) * ThPd * cos(ThP);
H2(1,2) = -R2*sin(Th0+ThR0+ThR1+ThR2);
H2(1,1) = H2(1,2) - R1*sin(Th0+ThR0+ThR1);
H2(2,2) = R2*cos(Th0+ThR0+ThR1+ThR2);
H2(2,1) = H2(2,2) + R1*cos(Th0+ThR0+ThR1);
H1(1,1) = H2(1,1) - R0*sin(Th0+ThR0);
H1(2,1) = H2(2,1) + R0*cos(Th0+ThR0);
Thd = inv(H2) * ([Pxd; Pyd] - H1*Th0d);
% Angle rates
ThR1d = Thd(1);
ThR2d = Thd(2);
% Save for plotting
QdotRef(4) = ThR1d;
QdotRef(5) = ThR2d;

% [Pxdd; Pydd] = [H1dot]*Th0d+[H1]*Th0dd + [H2dot]*Thd+[H2]*Thdd
Pxdd = Xcdd - (LP - LcP)*(ThPdd*sin(ThP) + ThPd^2*cos(ThP));
Pydd = Ycdd + (LP - LcP)*(ThPdd*cos(ThP) - ThPd^2*sin(ThP));
H2dot(1,2) = -R2*(Th0d+ThR1d+ThR2d)*cos(Th0+ThR0+ThR1+ThR2);
H2dot(1,1) = H2dot(1,2) - R1*(Th0d+ThR1d)*cos(Th0+ThR0+ThR1);
H2dot(2,2) = -R2*(Th0d+ThR1d+ThR2d)*sin(Th0+ThR0+ThR1+ThR2);
H2dot(2,1) = H2dot(2,2) - R1*(Th0d+ThR1d)*sin(Th0+ThR0+ThR1);
H1dot(1,1) = H2dot(1,1) - R0*Th0d*cos(Th0+ThR0);
H1dot(2,1) = H2dot(2,1) - R0*Th0d*sin(Th0+ThR0);
Thdd = inv(H2)*([Pxdd; Pydd]-H2dot*Thd-[H1dot]*Th0d-[H1]*Th0dd);

```

```
% Angle accelerations
ThR1dd = Thdd(1);
ThR2dd = Thdd(2);
QddotRef(4) = ThR1dd;
QddotRef(5) = ThR2dd;

%% %%%%%%%%%%%%%%
%% Find needed control wheel torque %%
%% %%%%%%%%%%%%%%
Qdot = QR.*f;
Qdot = QdotRef;
Qddot = QddotRef;

[|Ls,Hdots]| = AngMo2(Ls,Ms,CMs,ls,Q,Qdot,Qddot);

%% Cost Function, Jopt
%% Wheel torque is the change in total angular momentum
%% Jopt1 is integrated while Jopt 2 is not
Jopt1 = abs(Hdots(7));
Jopt2 = Jopt1;
```

## REFERENCES

1. Robert E. Lindberg, Richard W. Longman, and Michael F. Zedd, "Kinematic and Dynamic Properties of an Elbow Manipulator Mounted on a Satellite," *The Journal of the Astronautical Sciences*, Vol. 38, No. 4, October-December 1990, pp. 397-421.
2. P. K. C. Wang, "Control Strategy for a Dual-Arm Maneuverable Space Robot," *Proceedings of the Workshop on Space Telerobotics*, Vol 2, 1987, pp. 257-266.
3. Richard W. Longman, Robert E. Lindberg, and Michael F. Zedd, "Satellite-Mounted Robot Manipulators -- New Kinematics and Reaction Moment Compensation," *The International Journal of Robotics Research*, Vol. 6, No. 3, pp. 87-103, Fall 1987.
4. Z. Vafa and S. Dubowsky, "On the Dynamics of Manipulators in Space Using the Virtual Manipulator Approach," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, NC, 1987, pp. 579-585.
5. Richard W. Longman, "The Kinetics and Workspace of a Satellite-Mounted Robot," *The Journal of the Astronautical Sciences*, Vol. 38, No. 4, October-December 1990, pp. 423-441.
6. Yoshihiko Nakamura and Ranjan Mukherjee, "Nonholonomic Path Planning of Space Robots via Bi-Directional Approach," *1990 IEEE International Conference of Robotics and Automation*, Cincinnati, Ohio, pp. 1764-1769.
7. Z. Vafa, "Space Manipulator Motions with No Satellite Attitude Disturbances," *Proceedings of 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, 1990, pp. 1770-1775.
8. C. L. Chung, S. Desa, and C. W. deSilva, "Base Reaction Optimization of Redundant Manipulators for Space Applications," Report No. NASA-CR-186274, 1988.
9. Z. Vafa and S. Dubowsky, "On the Dynamics of Space Manipulators Using the Virtual Manipulator, with Applications to Path Planning," *Journal of the Astronautical Sciences*, Vol. 38, No. 4, Oct-Dec 1990, pp. 441-472.
10. C. C. Nguyen, F. J. Pooran, and T. Premack "Trajectory Control of Robot Manipulators with Closed-Kinematic Chain Mechanism," Report No. NASA-TM-89305, 1987.
11. Y. Hu and A. Goldenberg, "An Adaptive Approach to Motion and Force Control of Multiple Coordinated Robot Arms," *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, 1989, pp. 1091-1096.

12. Michael W. Walker, Dongmin Kim, and Joseph Dionise, "Adaptive Coordinated Motion Control of Two Manipulator Arms," *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, 1989, pp. 1084-1090.
13. K. Yoshida, R. Kurazume, and Y. Umetani, "Torque Optimization Control in Space Robots with a Redundant Arm," *Proceedings of IROS '91*, Osaka, Japan, 1991, pp. 1647-1652.
14. Shaheen Ahmad and Mohamed Zribi, "Lyapunov Based Control Design for Multiple Robots Handling a Common Object," *Mechanics and Control: Proceedings of the 4th Workshop on Mechanics and Control*, University of Southern California, Los Angeles, Jan. 21-23, 1991, edited by J.M. Skowronski, H. Flashner, and R.S. Guttail, pp. 1-17, Springer-Verlag, 1992.
15. Stanley A. Schneider and Robert H. Cannon Jr., "Object Impedance Control for Cooperative Manipulation: Theory and Experimental Results," *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, Arizona, 1989, pp. 1076-1083.
16. Discussions between John L. Junkins, George J. Eppright Chair Professor at Texas A&M University and visiting Co-Chair of the Space Systems Academic Group at the Naval Postgraduate School and the author, July-August 1992.
17. Watkins Jr., R.J., *The Attitude Control of Flexible Spacecraft*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1991.
18. Hailey, Jeffrey A., *Experimental Verification of Attitude Control Techniques for Flexible Spacecraft Slew Maneuvers*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1992.
19. Sorenson, Dennis, *Design and Control of a Space Based Two Link Manipulator with Lyapunov Based Control Laws*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1992.

## INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3. Chairman, Code AA Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	1
4. Chairman, Code SP Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	1
5. Professor Brij N. Agrawal, Code AA/Ag Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	2
6. Professor Donald A. Danielson, Code Dd Department of Mathematicss Naval Postgraduate School Monterey, California 93943-5002	1
7. Professor I. Michael Ross, Code AA/Ro Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	1
8. Professor Harold A. Titus, Code EC/Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1

- |     |   |   |
|-----|---|---|
| 9.  | Dr. Hyochoong Bang, Code AA/Ba<br>Department of Aeronautics and Astronautics<br>Naval Postgraduate School<br>Monterey, California 93943-5002        | 1 |
| 10. | Professor John L. Junkins<br>Department of Aerospace Engineering<br>H. R. Bright Building<br>Texas A&M University<br>College Station, TX 77843-3141 | 1 |
| 11. | Capt Gary E. Yale<br>10801 Cordova NE<br>Albuquerque, NM 87112  | 2 |